Alexis Balayre

Future Position Prediction for Pressure Refuelling Port of Commercial Aircraft

School of Aerospace, Transport and Manufacturing
Computational and Software Techniques in Engineering

MSc
Academic Year: 2023–2024

Supervisors: Dr Boyu Kuang and Dr Stuart Barnes
May 2024

School of Aerospace, Transport and Manufacturing
Computational and Software Techniques in Engineering

MSc

Academic Year: 2023–2024

Alexis Balayre

Future Position Prediction for Pressure Refuelling Port of Commercial Aircraft

Supervisors: Dr Boyu Kuang and Dr Stuart Barnes
May 2024

This thesis is submitted in partial fulfilment of the requirements for the degree of MSc.

Academic Integrity Declaration

I declare that:

- the thesis submitted has been written by me alone.
- the thesis submitted has not been previously submitted to this university or any other.
- that all content, including primary and/or secondary data, is true to the best of my knowledge.
- that all quotations and references have been duly acknowledged according to the requirements of academic research.

I understand that to knowingly submit work in violation of the above statement will be considered by examiners as academic misconduct.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ML	Machine Learning
DL	Deep Learning
AI	Artificial Intelligence
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
EKF	Extended Kalman Filter
AAGR	Autonomous Aircraft Ground Refueling
AGR	Aircraft Ground Refueling
UAV	Unmanned Aerial Vehicle
AAR	Autonomous Aerial Refueling
DGPS	Differential Global Positioning System
SVM	Support Vector Machine
HOG	Histogram of Oriented Gradients
SOTA	State-of-the-Art
AIS	Automatic Identification System
GPS	Global Positioning System
IoU	Intersection over Union
TP	True Positive
FP	False Positive
FN	False Negative
TN	True Negative
AP	Average Precision
AR	Average Recall
SSD	Single Shot Multibox Detector
YOLO	You Only Look Once
IoT	Internet of Things
AFRL	Air Force Research Laboratory
AR3P	Autonomous & Robotic Remote Refueling Point
BIPRS	Brightness Invariant Port Recognition System

Chapter 1

Introduction

1.1 Background and Motivation

Ground pressure refuelling is a standard method used to refuel commercial aircraft safely and efficiently. This process involves using a system of underground fuel pipelines and hydrants at aircraft parking spots [4]. When an aircraft is ready for refueling, a hydrant dispenser vehicle connects to the hydrant pit and delivers fuel to the aircraft through a flexible hose [44] (see Figure 1.1). This method allows for high fuel flow rates and significantly reduces aircraft turnaround times [4]. However, this process also presents several challenges, particularly in terms of safety and accuracy. In the past, there have been several incidents involving ground pressure refueling, including fuel spills, overfills, and equipment failures [37, 10]. Fortunately, with the advancement of technology, many of these challenges will be addressed, and the refueling process will become safer and more efficient.

Figure 1.1: Pressure Refuelling of a Commercial Aircraft. Source: Tom Boon/Simple Flying

The aviation industry is undergoing a significant transformation with the development of the airport of the future, commonly known as ‘Smart Airport’ or, more recently, ‘Airport 4.0’. The concept of Smart Airport encompasses the use of cutting-edge information technologies, such as the Internet of Things (IoT), Artificial Intelligence (AI), and Blockchain, to monitor, analyse, and integrate real-time data on the airport’s status. This integration aims to achieve optimal operational efficiency and enhance the quality of service. Taking the concept a step further, Airport 4.0 envisages an airport driven entirely by AI, capable of making autonomous decisions thanks to self-learning mechanisms. This advance aims to automatically predict and

manage various airport scenarios, making it easier to automate numerous processes. The result is a substantial reduction in operating costs and error rates [34].

Among these, automated refuelling systems play a crucial role in ensuring efficient and accurate refuelling of aircraft. However, one of the main challenges of this automation process is the accurate detection of the aircraft's refuelling port, which is relatively small and can easily be obscured by other visual elements on or near an aircraft. For example in the Figure 1.2, the refuelling port is located on the wing of the aircraft and can be difficult to detect due to motion blur, occlusion, or being out of view. This challenge is further compounded by the fact that aircraft refuelling ports can vary in size, shape, and location depending on the aircraft type and manufacturer. Scanning the entire area of each video frame is both time-consuming and inaccurate. It is therefore essential to develop a more efficient and accurate method of locating the refuelling port.

(a) Motion Blur Example (b) Occlusion Example (c) Out-of-View Example

Figure 1.2: Challenges in Detecting Aircraft Refuelling Port

1.2 Research Gap

Despite significant advancements in autonomous aircraft ground refuelling technologies, critical challenges remain, particularly in the accurate detection and positioning of the refuelling port. The small size and varied locations of refuelling ports, often obscured by visual elements like motion blur and occlusions, complicate this task. Existing systems have made progress using machine vision, but they are limited by inefficiencies in scanning entire video frames and inaccuracies under different environmental conditions. Furthermore, while current methodologies leveraging convolutional neural networks (CNNs) and Kalman filters have improved detection accuracy, they still struggle with real-time performance and adaptability in dynamic environments. The robustness of these systems in varied lighting conditions and their capability to handle different refuelling port types and obstructions need enhancement. Additionally, the application of advanced deep learning models like Long Short-Term Memory (LSTM) networks, Recurrent Neural Networks (RNNs), and Transformers in this context is underexplored.

1.3 Aim and Objectives

This thesis aims to address the previous research gaps by developing a framework for predicting the future position of commercial aircraft refuelling ports using advanced Object Detection models and Deep Learning to leverage the spatial-temporal relationship between frames in a video.

1. Conduct a comprehensive review of state-of-the-art methods for Object Detection, Object Tracking, and Deep Learning Sequence Models.

2. Annotate and preprocess video datasets of aircraft refuelling ports to ensure high-quality training and testing data.

3. Design and develop a framework for accurately tracking and predicting the future position of aircraft refuelling ports.

1.4 Technological Contributions

This thesis makes several technological contributions aimed at advancing the field of automated aircraft refuelling systems. The primary contribution is the integration of advanced deep learning models, including Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) networks, to predict the future positions of refuelling ports by leveraging their superior spatio-temporal data processing capabilities. Another significant contribution is the development of an innovative framework that combines object detection, target tracking, and future position prediction to enhance the accuracy and efficiency of automated refuelling systems. This framework is designed to handle the small pixel ratio of refuelling ports efficiently, thus optimising the detection and tracking process. The use of Extended Kalman Filtering (EKF) further refines predictions, ensuring real-time adaptability and accuracy, which is crucial for practical applications in busy airport environments.

1.5 Thesis Layout

The following sections of this thesis provide a detailed exploration and analysis of the methodologies, experiments, and findings related to the development of an advanced framework for predicting the future positions of aircraft refuelling ports. The Chapter 2 (Literature Review) presents an overview of the current state-of-the-art methods in automated aircraft refuelling systems, object detection, and deep learning for spatio-temporal prediction. The Chapter 3 (Methodology) describes the step-by-step approach taken in dataset preparation, framework design, and model training. The Chapter 4 (Experiment Design) outlines the experimental setups and comparison studies conducted to evaluate the performance of the proposed models. In the Chapter 5 (Results and Discussion), the outcomes of these experiments are presented and analysed, offering insights into the effectiveness and implications of the research findings. Finally, the Chapter 6 (Conclusion and Future Work) summarises the key contributions of the thesis, reflects on the significance of the results, and proposes directions for future research to further advance the field of automated aircraft refuelling systems.

Chapter 2

Literature Review

2.1 Automated Refulling Systems in the Aviation Industry

Ground refueling operations are essential to maintaining aircraft availability and operational efficiency. The transition from manual to automated systems is designed to improve the safety, efficiency and reliability of these operations [38]. The concept of Autonomous Aircraft Ground Refueling (AAGR) emerged in the 1980s in the United States to address the US Air Force's need to protect ground personnel from potential threats during refueling operations [46]. In the early 1990s, Bennett et al. [3] introduced the Brightness Invariant Port Recognition System (BIPRS), marking a significant advancement in machine vision systems for identifying aircraft refuelling ports. In 2010, the Air Force Research Laboratory (AFRL) showcased the world's first Automated Aircraft Ground Refuelling system prototype through a video demonstration. This system featured a robot equipped with a fuel nozzle and a single-point refuelling adapter, enabling autonomous engagement with the aircraft's refuelling panel, as illustrated in Figure 2.1 [6]. This project will give birth to the Autonomous & Robotic Remote Refuelling Point (AR3P) project.

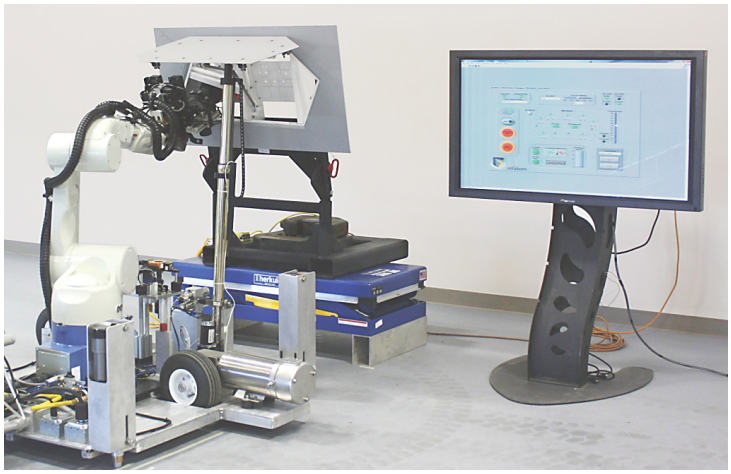


Figure 2.1: AFRL's Automated Aircraft Ground Refueling system prototype robot (Photo Credit: AFRL/RXQ Robotics Group)

The Autonomous & Robotic Remote Refuelling Point (AR3P) project, developed by the U.S. Army, represents a pioneering initiative in unmanned refuelling operations for rotary-wing aircraft. This project leverages advanced robotics, including self-aligning mechanisms

and articulated arms equipped with sensors, to facilitate rapid and safe refuelling processes on non-contiguous battlefields. The AR3P system minimises the time aircraft spend on the ground and enhances safety by reducing soldier exposure at fueling stations. Initially demonstrated in a Limited Initial Capabilities event, the AR3P aims to meet the evolving range and endurance requirements of Army Aviation. The project integrates existing technologies with novel systems designed in-house, supported by commercial off-the-shelf components and additive manufacturing. Currently, AR3P is progressing through its development phases, addressing technical risks, and preparing for further testing and eventual deployment, as shown in Figure 2.2 [15].

Figure 2.2: AR3P Concept Development Prototype Robot (Photo Credit: U.S. Army)

The AR3P project exemplifies the intersection of advanced robotics and practical military applications, highlighting the potential of automated systems to transform operational paradigms. Figure 2.3 provides visual insights into the capabilities of the AR3P system, by performing autonomous hot refuelling. During this test, the robot is equipped with a LIDAR sensor and a camera to detect the aircraft's refuelling port. In Figure 2.3a, the AR3P robot is seen approaching a detected aircraft, demonstrating its autonomous navigation and alignment capabilities. In Figure 2.3b, the AR3P robot is shown engaging the aircraft refuelling port, emphasising its precision and functionality in connecting to the aircraft's fuel port. These images illustrate the practical implementation of robotic technologies in enhancing the safety, efficiency, and speed of refuelling operations, particularly in challenging and hazardous environments [2].

(a) AR3P Robot Approaching Detected Aircraft (Photo Credit: Stratom)

(b) AR3P Robot Engaging Aircraft Refueling Port (Photo Credit: Stratom)

Figure 2.3: AR3P Robot Hot Refueling Demonstration for S-70 Helicopter

Unfortunately, there is very little literature on existing AAGR systems, as most research is carried out by the military and is classified. The most recent papers cover Autonomous Aerial Refueling (AAR) systems, which are used to refuel unmanned aerial vehicles (UAVs) in mid-air. These systems are designed to extend the flight time and range of UAVs by enabling them to refuel without landing. AAR systems are particularly challenging due to the high speeds and altitudes involved, as well as the need for precise measurement and tracking of the relative position between the receiver aircraft and the tanker aircraft are critical, particularly during the docking phase (see figure 2.4) [20, 53]. Zhong et al. [62] propose a robust solution that utilises monocular vision combined with an extended Kalman filter (EKF) to address this challenge. By implementing EKF, the system can provide reliable position estimations and track the drogue within a specified region of interest (ROI), even in the presence of disturbances such as air turbulence. As shown in figure 2.5, this system initialises the state and covariance matrices, predicts the drogue's position, updates the state based on new measurements, and continuously refines the ROI for subsequent image processing. This approach significantly reduces the processing time and improves the detection frequency from 10 Hz to up to 30 Hz by focusing computational resources on the predicted ROI.

Figure 2.4: Autonomous Aerial Refueling (AAR) of X-47B Unmanned Combat Air System Demonstrator (Photo Credit: U.S. Navy)

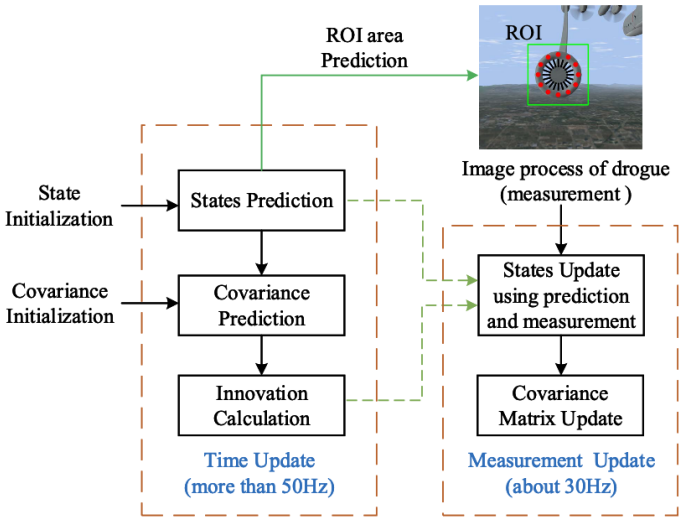


Figure 2.5: Autonomous Air Refueling Detection System with EKF. Source: Zhong et al. [62]

Recent advancements in autonomous ground refuelling have been driven by improvements in computer vision and robotics. Yildirim et al. [58] presented the PosEst system, which combines 2D RGB images with 3D point cloud data to enhance detection accuracy. This system uses a custom-trained EfficientNet-B0 CNN for object detection and leverages the Kalman filter for stable 3D pose estimation (see Figure 2.6). The PosEst method employs a dual approach of high-precision detection and robust tracking. By predicting and updating the object's state in real-time, the Kalman filter facilitates continuous and precise alignment of the fuel nozzle with the refuelling adaptor, even in dynamic environments. This approach significantly reduces the risks associated with manual refuelling and improves operational efficiency and safety.

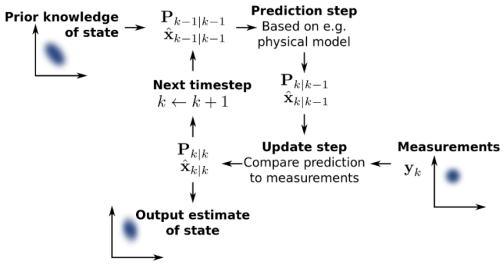


Figure 2.6: Kalman Filter Workflow for Pose Estimation in Autonomous Ground Refueling. Source: Yildirim et al. [58]

One of the primary challenges in AAGR is the accurate detection and positioning of the refuelling port under varying environmental conditions. Robust datasets for scene recognition and machine learning applications have been developed to address these challenges. Kuang et al. [29] introduced a comprehensive dataset for AAGR, addressing significant challenges such as variant illumination conditions, different refuelling port types, and environmental obstructions. The dataset comprises over 26,000 labeled images collected through image crawling from 13 different databases, followed by augmentation to ensure diversity (see Figure 2.7). Additionally, recent innovations have introduced hybrid datasets combining real and synthetic data for training and validating systems [59]. This approach offers a wide range of scenarios and conditions, improving the robustness and accuracy of automated refuelling systems. The development of high-quality datasets is pivotal in improving the robustness and reliability of AAGR systems.

Figure 2.7: AAGR Dataset Overview. Source: Kuang et al. [29]

2.2 Object Detection and Tracking in Computer Vision

In Computer Vision, Object Detection refers to the identification and location of individual objects within an image, providing both spatial information (bounding boxes) and confidence scores, which represent the probability that each detected object belongs to the predicted class [13]. For example, in the following image, there are five detections, including one 'ball' with a confidence level of 98% and four 'people' with confidence levels of 98%, 95%, 97% and 97%.

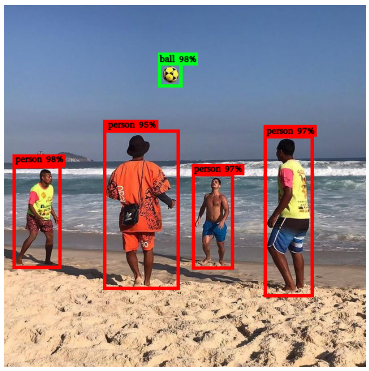


Figure 2.8: Example of outputs from an object detector [13].

Over the last few decades, Object Detection models based on Deep Learning have enjoyed remarkable success. These models fall into two main categories: two-stage detectors and single-stage detectors. On the one hand, two-stage detectors, such as R-CNN [19], Fast R-CNN [18], Faster R-CNN [42] and R-FCN [11], first generate region proposals and then refine these proposals into precise anchor boxes. While these models excel in detection accuracy, they typically suffer from large model sizes and slower detection speeds [27, 57]. On the other hand, single-stage detectors, including the SSD (Single Shot Multibox Detector) [36], YOLO (You Only Look Once) series [41, 40, 5, 8, 17, 25, 32, 26, 52, 55, 51, 56, 50], and RetinaNet [33] directly predict object locations and categories in a single network pass. These models are known for their high detection speeds but sometimes compromise accuracy [27, 57].

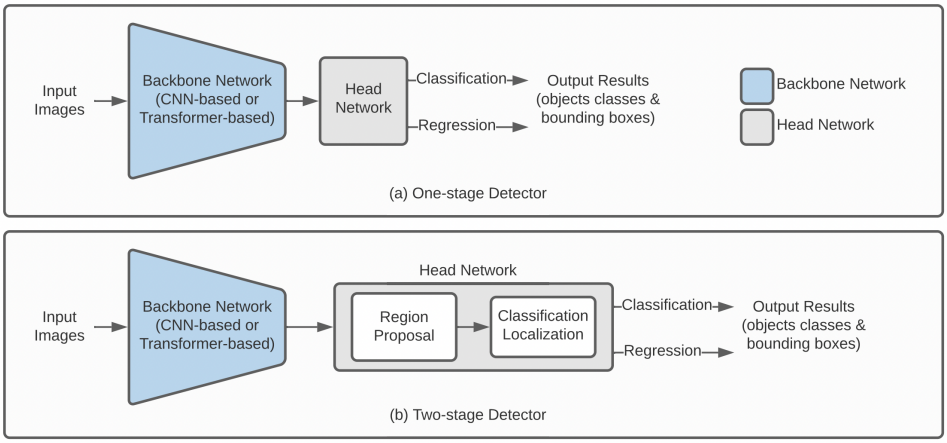


Figure 2.9: Basic deep learning-based one-stage vs two-stage object detection model architectures [27].

Evaluating Object Detection models involves several key metrics to measure their performance. One common metric is Intersection over Union (IoU), which measures the overlap between a predicted bounding box and a ground-truth bounding box, as shown in Figure 2.10.

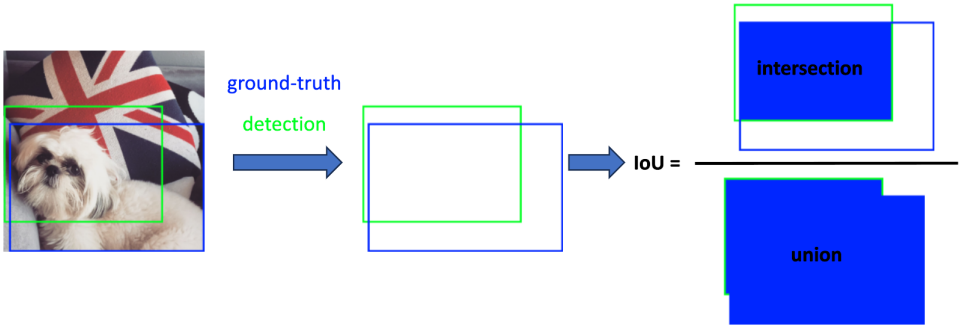


Figure 2.10: Intersection over Union (IoU) between a detection (in green) and ground-truth (in blue). [13]

Based on the IoU metric, a detection can be classified as a **True Positive (TP)** or a **False Positive (FP)** depending on whether the IoU value exceeds a certain threshold (T_{IoU}). If the IoU is above the threshold, the detection is considered correct (TP); otherwise, it is classified as a False Positive (FP). Additionally, **False Negatives (FN)** refer to ground truth objects not detected by the model, while **True Negatives (TN)** are correctly classified background detections [13]. These classifications allow for the calculation of the following metrics:

- **Precision**: The ratio of True Positives to the total number of detections, measuring the model's ability to avoid false positives:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.1}$$

- **Recall**: The ratio of True Positives to the total number of ground-truth objects, measuring the model's ability to avoid false negatives:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.2}$$

Common metrics used to evaluate Object Detection include:

- **Average Precision (AP)**: This combines precision and recall, providing a single figure summarizing the model's performance across different confidence thresholds. Common versions are AP@.5 (with a threshold of 0.5 IoU) and AP@[.5:.05:.95], which calculates the average of AP values over several IoU thresholds [13].
- **Average Recall (AR)**: This measures the recall of the model averaged across multiple IoU thresholds. It can be computed for different numbers of detections per image, such as AR@1, AR@10, etc. [13].
- **Inference Time**: The time taken by the model to process an image, which is critical for applications requiring real-time detection [13].
- **Model Size**: The number of parameters or the size of the model, affecting deployment, especially on devices with limited resources [13].

- **Efficiency**: This considers the trade-off between accuracy and speed, often visualized using the AP vs. inference time curve [13].

In addition to Object Detection, Object Tracking is another critical task in Computer Vision, involving the continuous monitoring of objects across video frames. Object Tracking methods can be broadly classified into two categories: **Generative Trackers** and **Discriminative Trackers** [7]. Generative trackers are capable of handling challenging scenarios such as occlusion and large-scale variation through particle sampling strategies, often integrated with various appearance models, including sparse representation and energy of motion. Discriminative trackers, by contrast, build robust classifiers using hand-crafted or deep features [7]. The combination of generative and discriminative approaches, as well as the integration of deep learning techniques such as fully revolutionary networks and Transformer models, has led to significant improvements in object detection performance [35, 61, 60]. In addition, the speed and computational requirements of these algorithms are critical factors influencing their practical applicability [61, 60, 30]. Advanced techniques in object tracking leverage both generative and discriminative models to amplify tracking efficacy. The utilisation of deep trackers has evidenced superior results on public tracking datasets, attributed to their potent feature extractors, accurate bounding box regressors, and discriminative classifiers [30]. Techniques such as deformable convolution and Transformer models extend traditional convolution or correlation methodologies to execute global feature matching, thereby enhancing tracking accuracy. The incorporation of contextual or knowledge information can substantially elevate performance, with methodologies like Particle Filtering, also recognised as Sequential Monte Carlo (SMC) methods, framed as problems of Bayesian inference in state space [9, 54]. The extended Kalman Filtering (EKF) is another advanced technique that has been employed to improve tracking accuracy by predicting the current status through the previous status and modifying the prediction result based on observation information [61, 60]. Despite these advancements, the integration of these methods in a complementary manner remains an open research area with substantial potential for advancing the field [35, 61].

2.3 Deep Learning for Spacio-Temporal Prediction

Time series prediction involves processing sequential data to predict future events or values. Various deep learning models have been applied to this task, requiring several preparatory steps such as collecting data, designating attribute types, dealing with inconsistencies and storing datasets. These datasets are usually classified into units of time such as seconds, minutes and hours, allowing the construction of metadata for machine learning [31].

Early Approaches and Models

The problem of predicting the future locations of objects has been extensively studied, particularly for static surveillance cameras. Initial efforts utilised recurrent neural networks (RNNs), including long-term memory networks (LSTMs) and gated recurrent units (GRUs), in an encoder-decoder format to encode past observations and decode future locations.

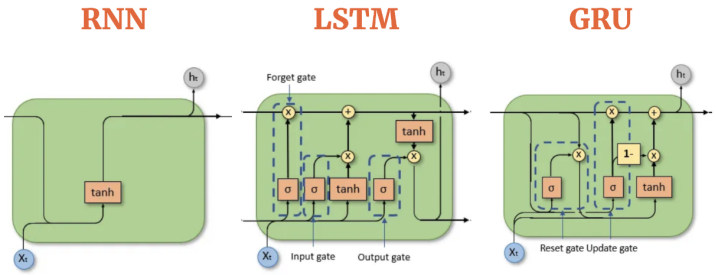


Figure 2.11: Comparing different Sequence models: RNN, LSTM, and GRU. Source: Colah's blog. Compiled by AIML.com

Early models integrated additional inputs such as environmental data and semantic actions to enhance prediction accuracy. For instance, Alahi et al. [1] proposed a Social-LSTM to model pedestrian trajectories and interactions, further improving global context capture through a social pooling module.

Transformer Models for Frame-Based Prediction

Transformer model, introduced by Vaswani et al. [49], has revolutionised sequence modeling with its efficient and powerful network structure. For video prediction, transformers leverage self-attention mechanisms to capture long-range dependencies across frames. The Hong et al. [21] study shows how transformers can effectively learn mobility patterns from historical frame sequences, achieving state-of-the-art performance in next-location prediction tasks.

Frame-Based Approaches in Spatio-Temporal Prediction

Video prediction, a critical task in spatio-temporal data analysis, involves forecasting future frames based on past and current frames. This process requires understanding both spatial and temporal dynamics within the data.

Convolutional LSTM (ConvLSTM)

ConvLSTM integrates convolutional operations into LSTM units to better capture spatial features in addition to temporal dependencies [47]. By processing video data as sequences of

frames, ConvLSTM effectively models the spatio-temporal dependencies necessary for accurate video prediction. Each frame serves as a spatial unit, and the sequence of frames provides temporal context, allowing the model to predict future frames by learning from the patterns in previous ones.

Cubic LSTM for Video Prediction

The Cubic Long Short-Term Memory (CubicLSTM) unit, as proposed by Fan et al. [14], extends the capabilities of ConvLSTM by separately processing spatial and temporal information through three branches: temporal, spatial, and output. This approach mitigates the computational burden and enhances prediction accuracy. Specifically:

- The *temporal branch* processes motion information by analysing the sequence of frames over time.
- The *spatial branch* captures object information within individual frames.
- The *output branch* combines temporal and spatial information to generate predicted future frames.

This separation of concerns allows the model to handle the complexities of video data more effectively, resulting in more accurate predictions of future frames.

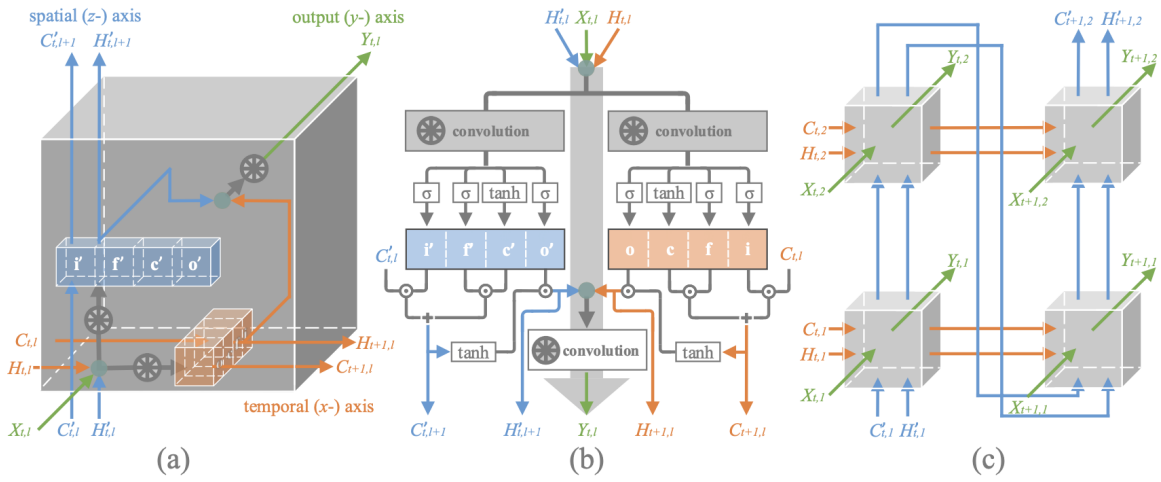


Figure 2.12: Cubic LSTM Architecture. (a) 3D structure of the CubicLSTM unit. (b) Topological diagram of the CubicLSTM unit. (c) Two-spatial-layer RNN composed of CubicLSTM units. The unit consists of three branches, a spatial (z-) branch for extracting and recognizing moving objects, a temporal (y-) branch for capturing and predicting motions, and an output (x-) branch for combining the first two branches to generate the predicted frames. Source: Fan et al. [14]

Unsupervised Video Forecasting with Flow Parsing

Jin et al. [24] introduced an unsupervised video forecasting approach that incorporates a flow parsing mechanism. This model separates the motion and appearance learning processes:

- The *motion stream* predicts optical flow between frames to capture dynamic changes.

- The *appearance stream* reconstructs frames to preserve spatial details.

By integrating these streams, the model can predict future frames with improved motion accuracy and appearance fidelity.

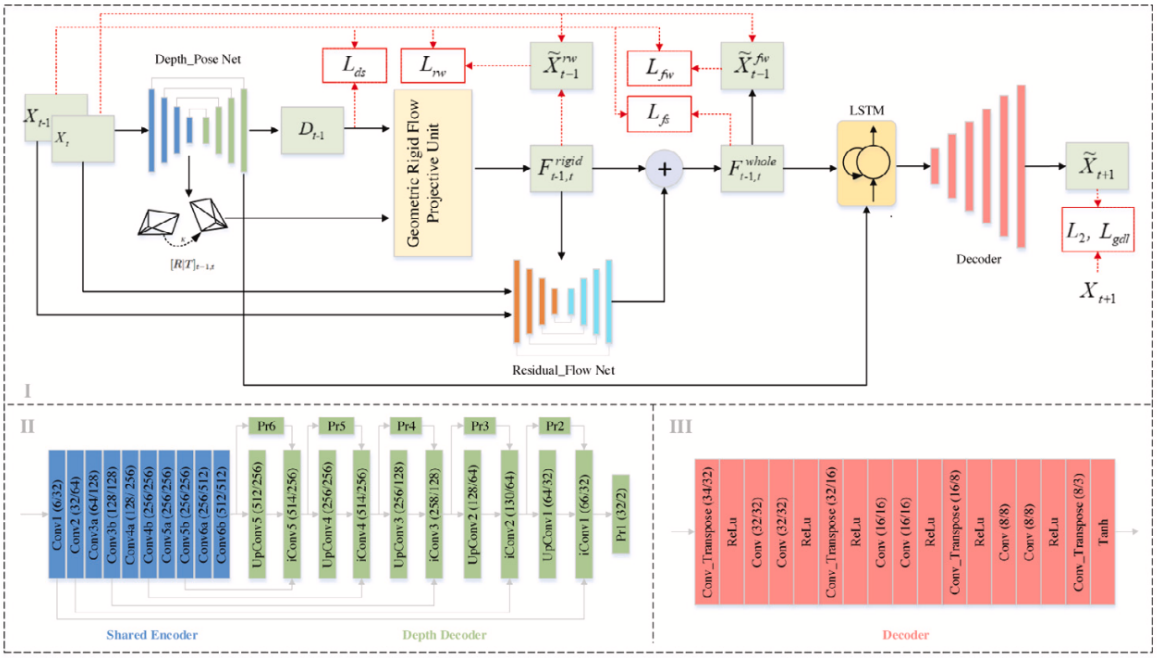


Figure 2.13: Model Architecture. Source: Jin et al. [24]

Joint Optimization with Synthesis and Optical Flow Estimation

Ranjan et al. [39] proposed a method that jointly optimizes frame synthesis and optical flow estimation. Their approach leverages:

- A *frame synthesis network* to generate predicted frames.
- An *optical flow estimation network* to capture motion dynamics between frames.

This joint optimization enhances the model's ability to produce accurate and visually coherent future frames.

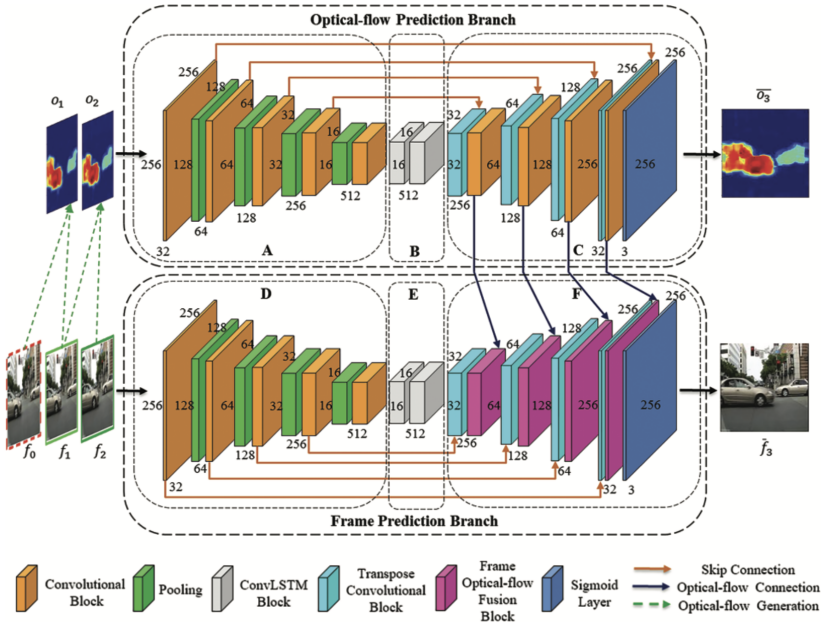


Figure 2.14: FPNet-OF model architecture. Source: Ranjan et al. [39]

Fusion-GRU

Karim et al. [28] developed the Fusion-Gated Recurrent Unit (Fusion-GRU) model to predict the future bounding boxes of traffic agents in risky driving scenarios. This model leverages multiple sources of information, including location-scale data, monocular depth information, and optical flow data, to capture complex interactions among information cues and transform them into hidden representations. The Fusion-GRU model uses an intermediary estimator and self-attention aggregation layer to enhance sequential dependencies for long-term predictions, demonstrating superior performance in challenging driving scenarios.

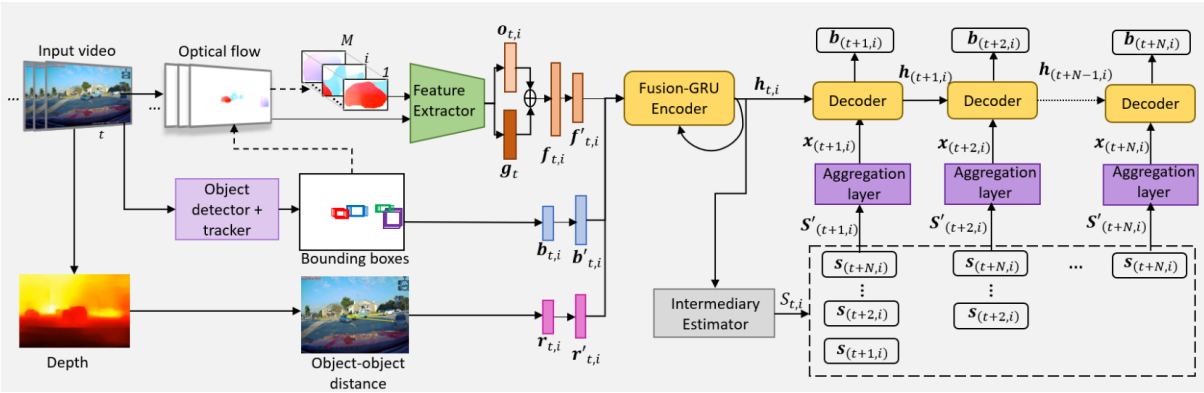


Figure 2.15: Fusion-GRU model architecture. Source: Karim et al. [28]

Dual-Branch Spatial-Temporal Learning Network

Huang and Guan [22] proposed a dual-branch video prediction network that aims to generate high-quality future frames by simultaneously capturing complex motion patterns and preserving appearance information. Their network includes two distinct units:

- The *motion prediction unit (MPU)* focuses on inter-frame motion and intra-frame appearance by using depth and multiple-scale convolutions, along with temporal attention mechanisms to enhance feature interactions over time.
- The *spatial prediction unit (SPU)* concentrates on spatial information, ensuring appearance consistency across video frames by capturing various appearance features.

This dual-branch approach addresses the limitations of relying on external information or complex state transition units, resulting in better visual quality and fewer blurry artifacts in predicted frames.

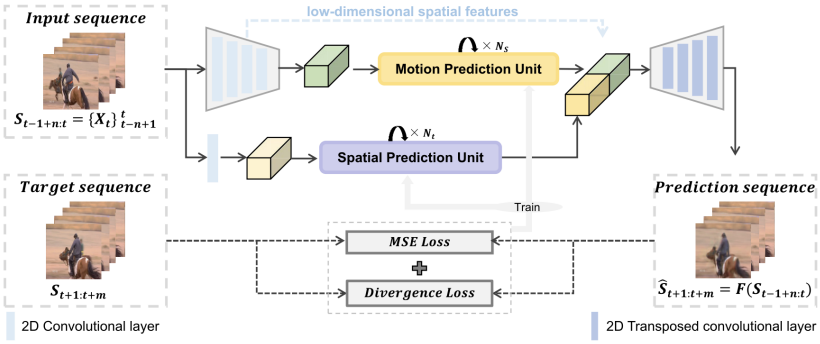


Figure 2.16: Dual-Branch Spatial-Temporal Learning Network. Source: Huang and Guan [22]

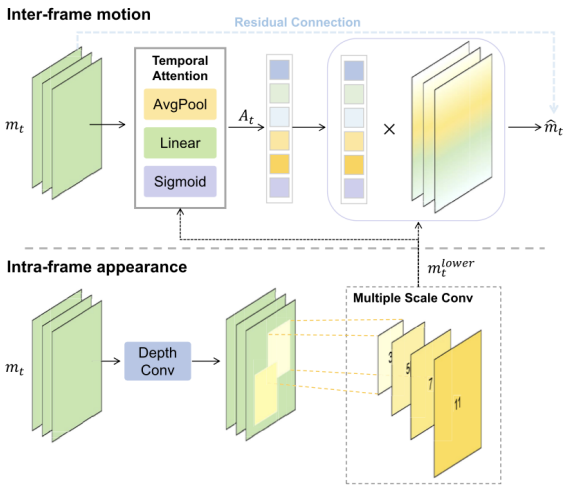


Figure 2.17: Dual-Branch Spatial-Temporal Learning Network. Source: Kang et al. [27]

Applications and Case Studies

In various applications, the frame-based approach to spatio-temporal prediction has proven highly effective:

Moving-MNIST Dataset

The Moving-MNIST dataset consists of sequences of moving handwritten digits. Each sequence includes a series of frames, and the goal is to predict future frames based on the observed

ones. This dataset contains 10000 videos, each consisting of 20 frames [48]. The CubicLSTM-based CubicRNN demonstrated superior accuracy compared to traditional ConvLSTM models, showcasing its ability to capture both motion and spatial features effectively [14].

Figure 2.18: 2-digit Moving MNIST data by Srivastava et al. [48]

Robotic Pushing Dataset

This dataset involves sequences of robotic arms pushing objects, with each frame representing a step in the sequence. The dataset contains 3456 training images with labels and 1024 validation images with labels [16]. The CubicLSTM model, combined with convolutional dynamic neural advection (CDNA) models, achieved higher accuracy and generated clearer frames compared to ConvLSTM and CNN-based models. This application highlights the importance of accurately predicting future frames to understand and anticipate the movements of robotic systems [14].

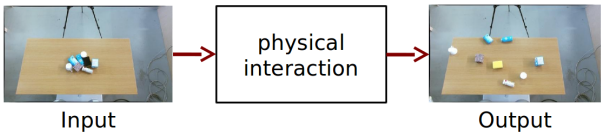


Figure 2.19: Robotic Pushing Dataset. Source: Eitel et al. [12]

KTH Action Dataset

The KTH Action dataset features videos of people performing various actions, with each video divided into frames. The dataset contains 2391 sequences [45]. The CubicLSTM model provided more accurate and visually consistent predictions of future frames compared to other state-of-the-art models like DrNet and MCnet. This demonstrates the model's effectiveness in understanding and predicting human actions over time [14].

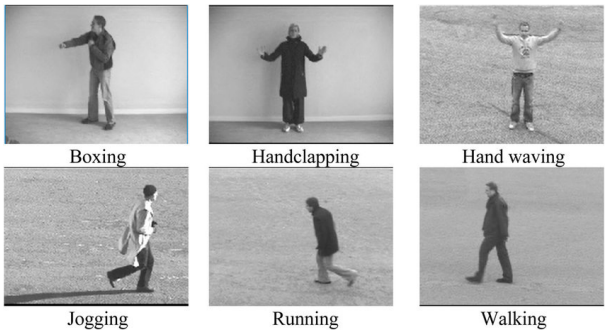


Figure 2.20: KTH Action Dataset. Source: Schuldt et al. [45]

UCF Sports Dataset

The UCF Sports dataset consists of a collection of footage collected from various sports that are typically shown on television channels such as the BBC and ESPN. It comprises a total of 150 sequences [43]. Huang and Guan's dual-branch network was tested on the UCF Sports dataset, which includes complex motion patterns and large movements. Their method outperformed other state-of-the-art methods in terms of visual quality and motion accuracy, as evidenced by superior metrics such as PSNR, SSIM, and LPIPS. The results validate the network's ability to handle intricate video prediction tasks without relying on additional information [22].

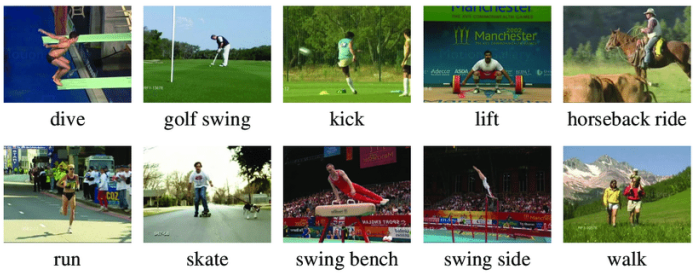


Figure 2.21: UCF Sports Dataset. Source: Rodriguez et al. [43]

Chapter 3

Methodology

3.1 Dataset Configuration

3.1.1 Provided Dataset Description

The 'Indoor Hangar Fueling Port Detection and Tracking Dataset' (HARD) is an integral part of Phase-1 of the ONEHeart project, funded by UKRI. The ONEHeart project aims to develop an automated aircraft refuelling system based on computer vision and robotics technology. This dataset can be utilised for various purposes, including but not limited to refueling port detection, fueling port tracking, camera pose estimation, and visual image processing. It is provided under the terms of the UKRI funding agreement. Unauthorised use, distribution, or reproduction is prohibited.

The HARD dataset consists of 21 video sequences captured in an indoor hangar at the Aerospace Integration Research Centre (AIRC). The videos were recorded using an Intel® RealSense™ D435 [23] (Shown in Figure 3.1), which provides depth information in addition to RGB data. The target area for detection and tracking is near the refueling port of an Airbus A320 wing. The videos were recorded under different lighting conditions, including indoor artificial lighting and natural daylight, to simulate real-world scenarios.

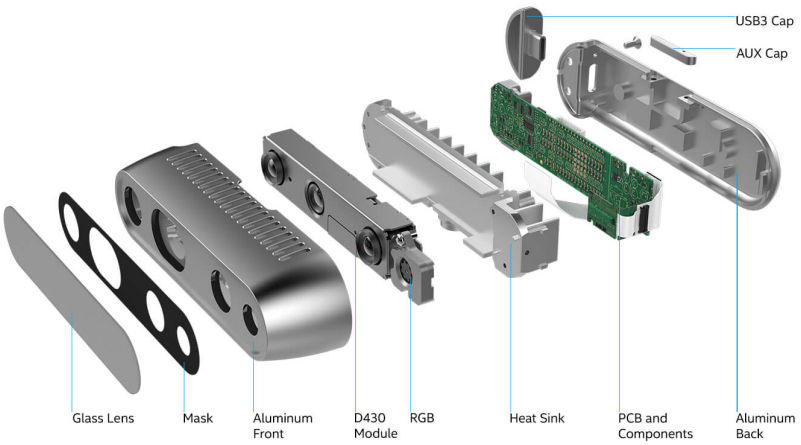


Figure 3.1: Intel® RealSense™ D435 Depth Camera. Source: Intel

There are three states of the refueling port in the dataset:

- **CLOSED**: The refueling port is closed.

- **OPEN**: The refueling port is open.
- **SEMI-OPEN**: The refueling port is partially open.

3.1.2 Data Annotation

The HARD dataset provided for this project was not fully annotated. Therefore, the first step in the data preparation process was to annotate the dataset. This annotation process was crucial to enable the training of machine learning models for accurate refueling port detection and tracking.

The annotation process involved several steps:

1. **Initial Manual Annotation**: Initially, 100 frames from each video sequence were manually annotated. This involved labeling the refueling port in each frame, which required identifying and marking the exact location of the refueling port using bounding boxes. This was done using the Label Studio tool, a powerful annotation platform that allows users to create and manage annotations in images and videos.

2. **Tool Used - Label Studio**: Label Studio was chosen for its flexibility and ease of use. It supports various annotation formats and integrates well with machine learning workflows. Users can draw bounding boxes around objects of interest, in this case, the refueling port, to create labeled datasets.

3. **Preliminary Dataset Creation**: The initial manual annotations created a preliminary dataset. This dataset was used to train a YOLOv10 (You Only Look Once, version 10) model for refueling port detection.

4. **Model Training**: The annotated dataset was used to train the YOLOv10 model. The model learned to detect the refueling port from the annotated images, improving its accuracy with each training iteration.

5. **Automated Annotation**: After training the YOLOv10 model, it was implemented as a backend service for Label Studio. This was deployed as a Docker container, allowing the model to be used for automated annotation of the remaining images in the dataset. The model predicted the location of the refueling port in new frames, and these predictions were used to annotate the rest of the dataset automatically.

6. **Review and Quality Control**: Each automatically generated annotation was reviewed manually to ensure accuracy. This review process involved verifying the location of the refueling port in each frame and making necessary corrections to the annotations. This thorough quality control ensured that the entire dataset was consistently and accurately labeled.

This comprehensive annotation process ensured that the entire dataset was accurately labeled, providing a robust foundation for training machine learning models aimed at refueling port detection and tracking. The combination of manual and automated annotation techniques maximised efficiency while maintaining high annotation quality.

3.1.3 Summary of Available Videos

Each video was assigned to a specific dataset (train, val, and test), with an approximate split of 70% for training, 15% for validation, and 15% for testing. The following table presents the available videos in the dataset along with the number of frames for each video and their assignment:

Type	Video Name	Number of Frames	Assignment
Closed	**video_lab_platform_1**	**624**	**train**
Closed	video_lab_platform_2	639	train
Closed	video_lab_platform_5	398	train
Closed	video_lab_platform_7	412	train
Closed	video_lab_platform_8	470	train
Closed	video_lab_platform_9	373	train
Closed	video_lab_manual_1	746	train
Closed	video_lab_platform_3	569	val
Closed	video_lab_platform_4	247	val
Closed	video_lab_platform_6	303	test
Closed	test_outdoor1	499	test
Open	**video_lab_open_1_____1**	**497**	**train**
Open	video_lab_open_1_____2	602	train
Open	video_lab_open_1_____3	313	train
Open	video_lab_open_1_____4	310	train
Open	test_indoor2	310	val
Open	test_indoor1	314	test
Semi-Open	**video_lab_semiopen_1_____1**	**739**	**train**
Semi-Open	video_lab_semiopen_1_____2	439	train
Semi-Open	video_lab_semiopen_1_____4	372	val
Semi-Open	video_lab_semiopen_1_____3	383	test

Table 3.1: Summary of available videos in the HARD dataset with their assignment.

3.1.4 Initial Data Distribution

Before balancing the data, the distribution of video frames across the different datasets (train, validation, and test) for each state of the fueling port (CLOSED, OPEN, and SEMI-OPEN) was as follows:

Type	Total Frames	Train	Test	Validation
CLOSED	5280	3662 (69.36%)	802 (15.19%)	816 (15.45%)
OPEN	2346	1722 (73.40%)	314 (13.38%)	310 (13.21%)
SEMI-OPEN	1933	1178 (60.94%)	383 (19.81%)	372 (19.24%)

Table 3.2: Distribution of frames across train, test, and validation sets for each state in the HARD dataset before balancing.

As shown in 3.2, the dataset initially had an imbalance in the number of frames for each state. For instance, the CLOSED state had significantly more frames compared to the OPEN

and SEMI-OPEN states. This imbalance could lead to biased training and inaccurate model performance, as the model might become overly familiar with the more prevalent CLOSED state and underperform on the less represented states.

3.1.5 Balanced Data Distribution

To address this issue, a balancing strategy was employed to create a more uniform dataset. The steps taken were as follows:

1. **Shuffling and Subsetting**: Each state (CLOSED, OPEN, and SEMI-OPEN) subset was shuffled, and only the minimum number of frames from each state was kept. This step ensured that the number of frames for each state was equal, avoiding bias toward any particular state.

2. **Merging Subsets**: The subsets were merged to create three balanced datasets: Training, Validation, and Test, each with an equal number of frames for each state.

3. **Final Shuffling and Resizing**: Each balanced dataset was shuffled again and resized to maintain the required split ratio of 70% for training, 15% for validation, and 15% for testing.

The balanced dataset will be used for training and evaluating the object detection model, while the full dataset will be utilised for sequence model training and framework evaluation.

The resulting distribution of frames across the train, test, and validation sets for each state after balancing is as follows:

Dataset	Total Frames
Train	3534 (69.57%)
Test	773 (15.22%)
Val	773 (15.22%)

Table 3.3: Distribution of frames across train, test, and validation sets for each state in the HARD dataset after balancing.

The primary reason for balancing the dataset is to ensure that the object detection model is equally trained on all states of the refueling port. An imbalanced dataset could cause the model to perform well on the more common states (like CLOSED) while underperforming on the less common states (like OPEN and SEMI-OPEN). By balancing the dataset:

- **Improved Generalization**: The model will have an equal opportunity to learn from all states, improving its generalization and robustness.

- **Avoiding Bias**: Equal representation of each state prevents the model from becoming biased towards any particular state.

- **Consistent Performance**: This strategy ensures consistent performance across all states, which is critical for the reliable operation of the automated refueling system.

3.1.6 Example Images from the Dataset

The following figures show annotated examples of the refueling port in different states:

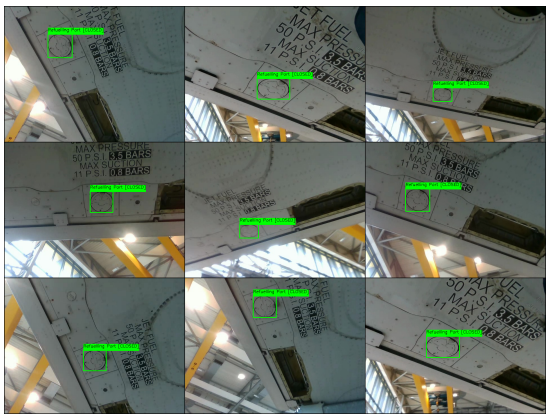


Figure 3.2: Annotated images of the refueling port in the CLOSED state.

Figure 3.3: Annotated images of the refueling port in the OPEN state.

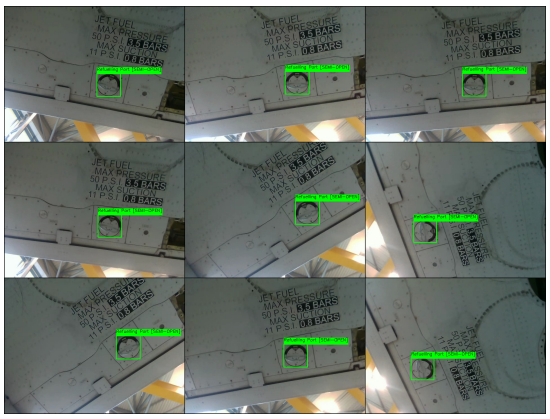
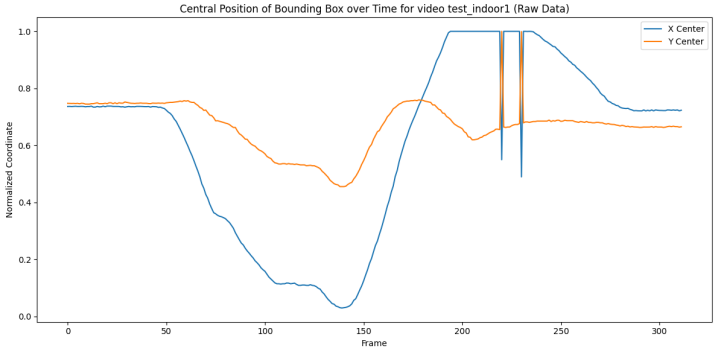
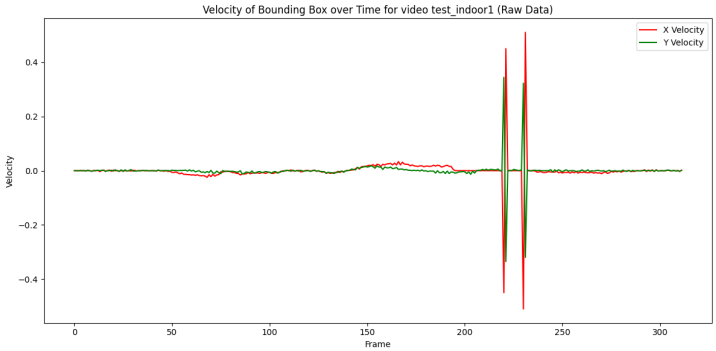


Figure 3.4: Annotated images of the refueling port in the SEMI-OPEN state.

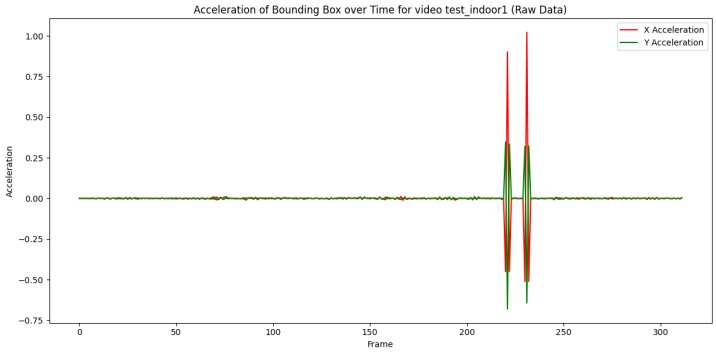
3.1.7 Sequence Model Data Preparation



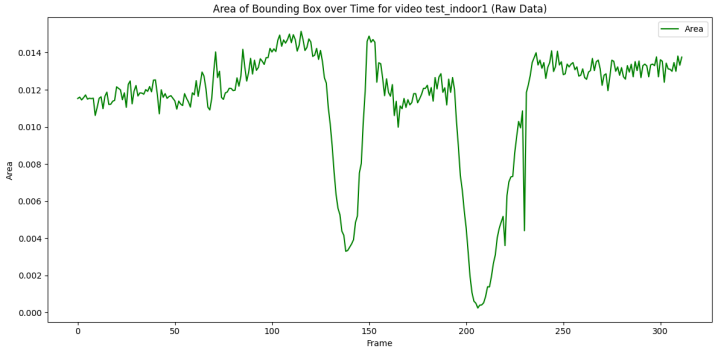
(a) Central position of the refueling port over time.



(b) Velocity of the refueling port over time.

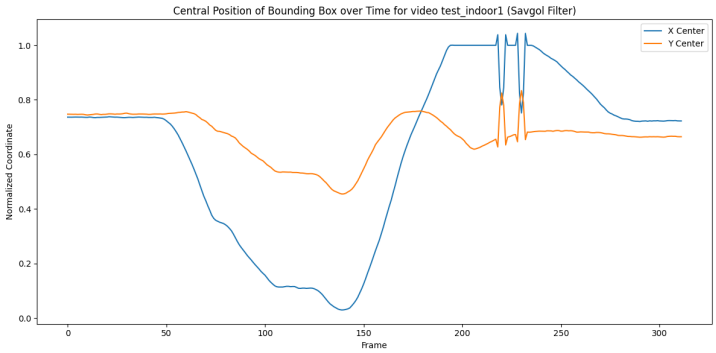


(c) Acceleration of the refueling port over time.

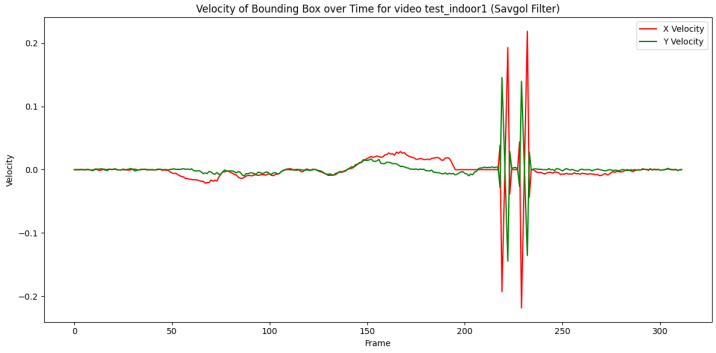


(d) Area of the refueling port over time.

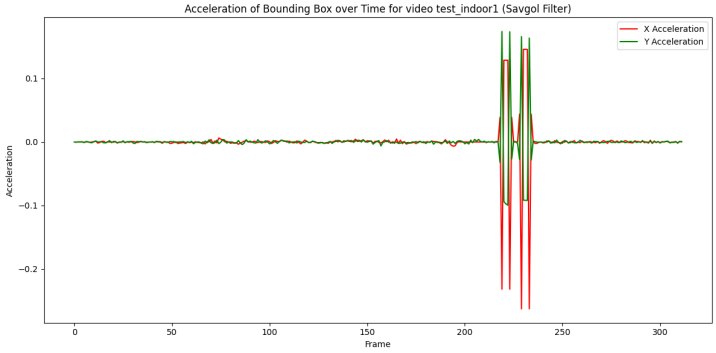
Figure 3.5: Temporal analysis of different metrics for the refueling port in the *test_indoor1* video. The metrics include (a) central position, (b) velocity, (c) acceleration, and (d) area, providing a comprehensive overview of the object's dynamics over time.



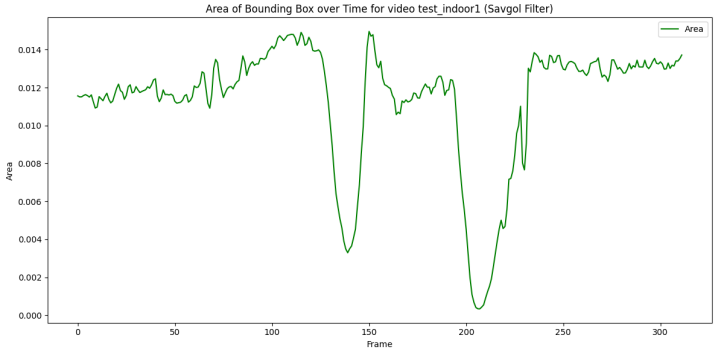
(a) Central position of the refueling port over time.



(b) Velocity of the refueling port over time.

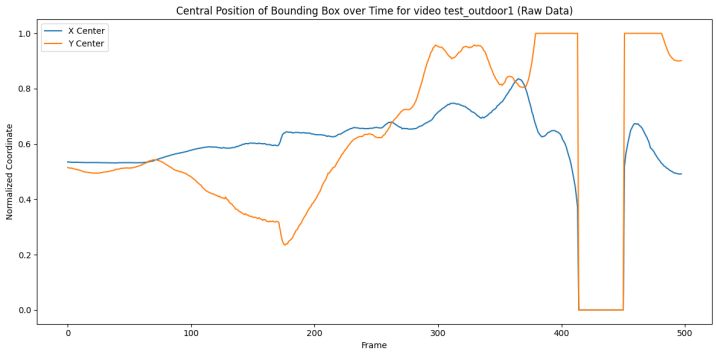


(c) Acceleration of the refueling port over time.

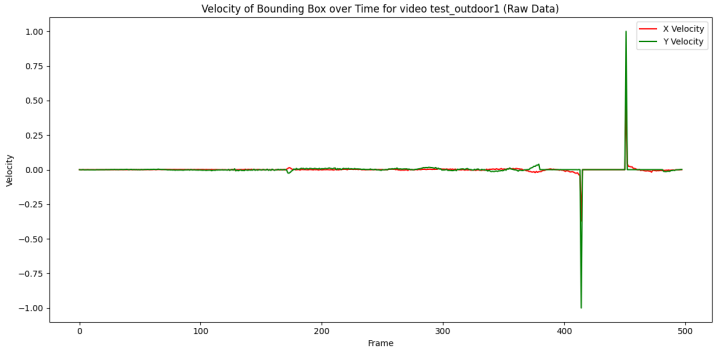


(d) Area of the refueling port over time.

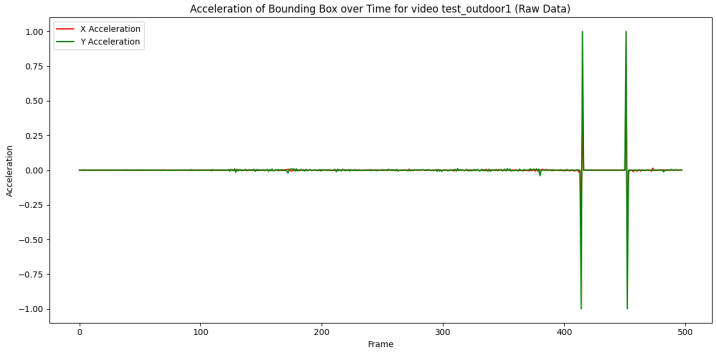
Figure 3.6: Temporal analysis of different metrics for the refueling port in the *test_indoor1* video. The metrics include (a) central position, (b) velocity, (c) acceleration, and (d) area, providing a comprehensive overview of the object's dynamics over time.



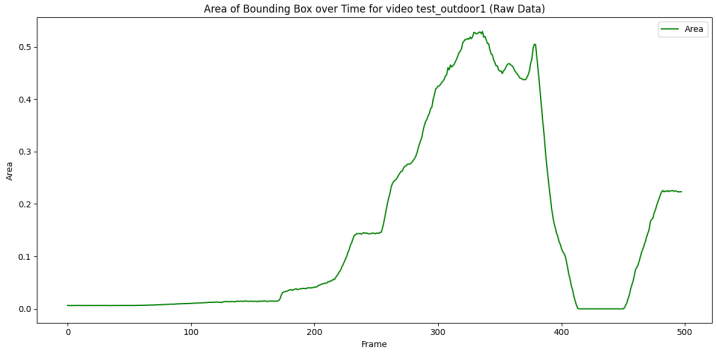
(a) Central position of the refueling port over time.



(b) Velocity of the refueling port over time.

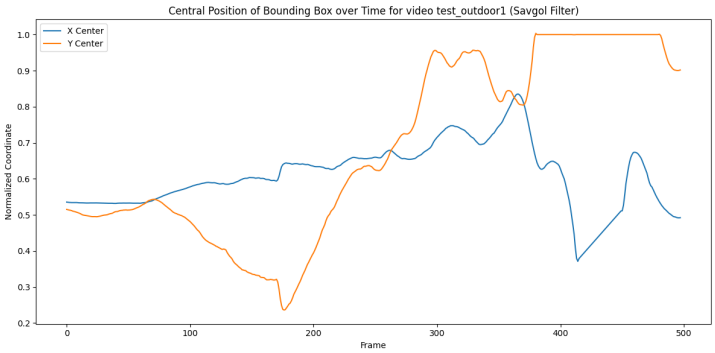


(c) Acceleration of the refueling port over time.

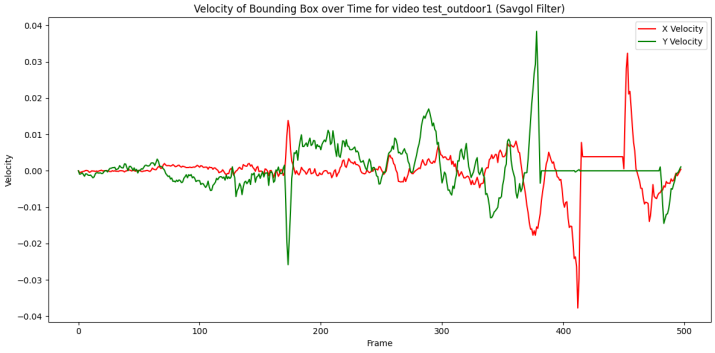


(d) Area of the refueling port over time.

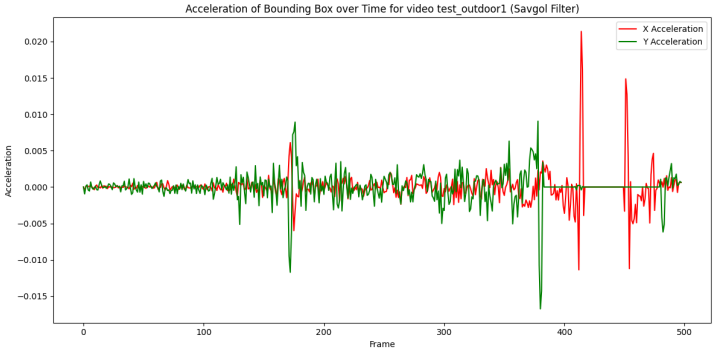
Figure 3.7: Temporal analysis of different metrics for the refueling port in the *test_outdoor1* video. The metrics include (a) central position, (b) velocity, (c) acceleration, and (d) area, providing a comprehensive overview of the object's dynamics over time.



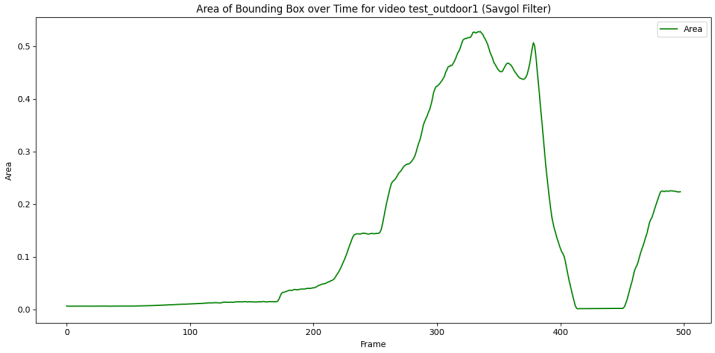
(a) Central position of the refueling port over time.



(b) Velocity of the refueling port over time.

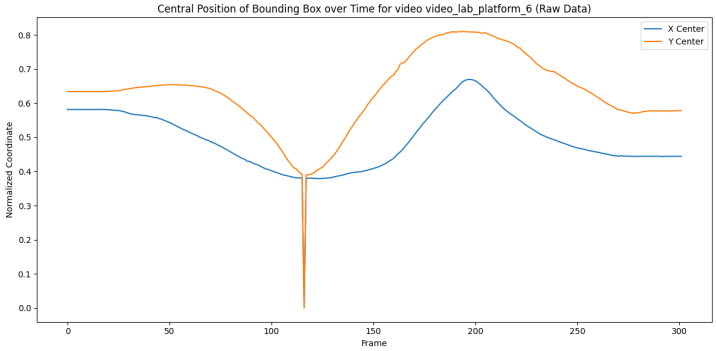


(c) Acceleration of the refueling port over time.

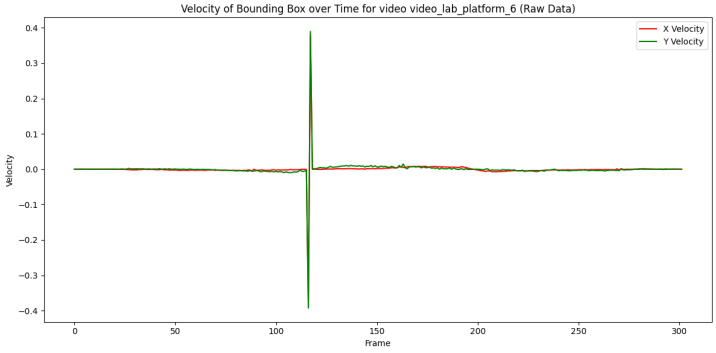


(d) Area of the refueling port over time.

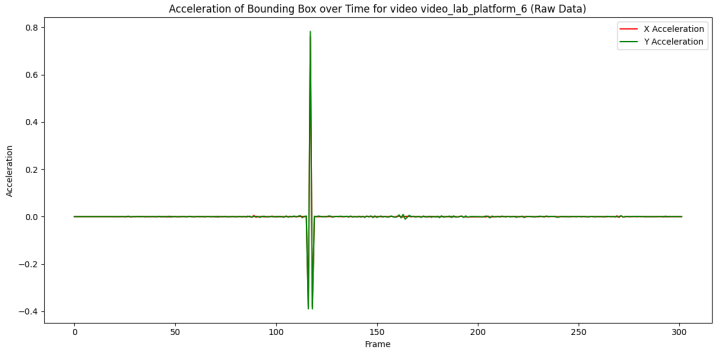
Figure 3.8: Temporal analysis of different metrics for the refueling port in the *test_outdoor1* video. The metrics include (a) central position, (b) velocity, (c) acceleration, and (d) area, providing a comprehensive overview of the object's dynamics over time.



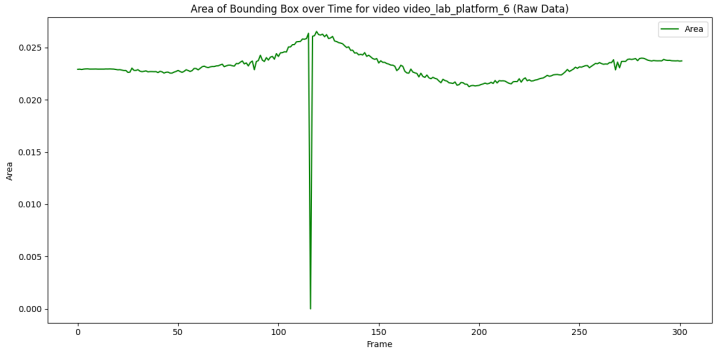
(a) Central position of the refueling port over time.



(b) Velocity of the refueling port over time.

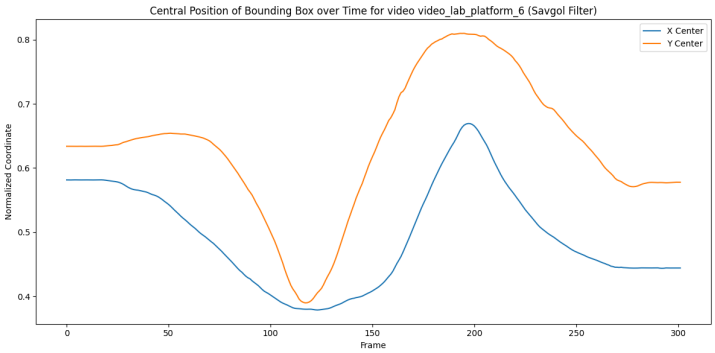


(c) Acceleration of the refueling port over time.

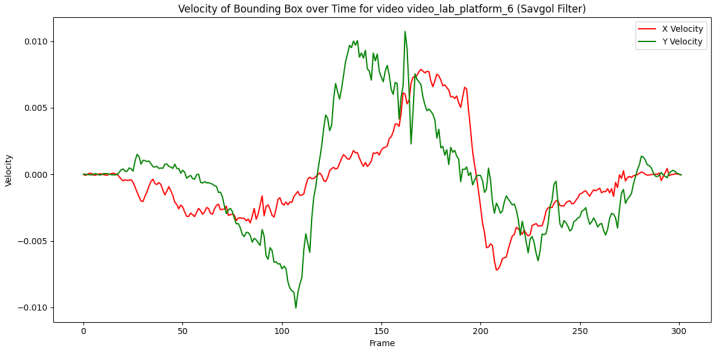


(d) Area of the refueling port over time.

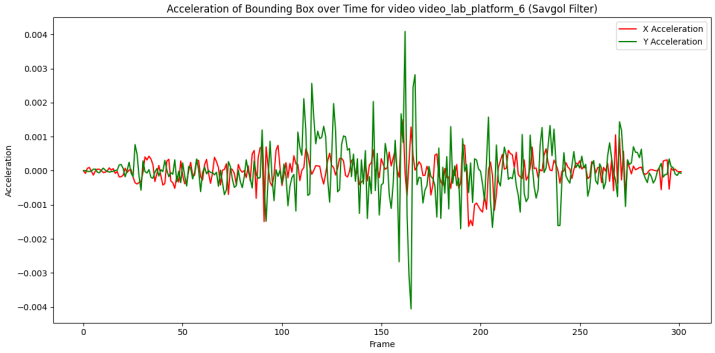
Figure 3.9: Temporal analysis of different metrics for the refueling port in the *test_video_lab_platform_6* video. The metrics include (a) central position, (b) velocity, (c) acceleration, and (d) area, providing a comprehensive overview of the object's dynamics over time.



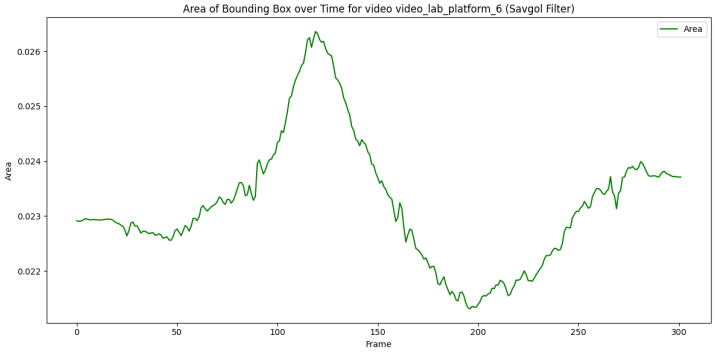
(a) Central position of the refueling port over time.



(b) Velocity of the refueling port over time.



(c) Acceleration of the refueling port over time.



(d) Area of the refueling port over time.

Figure 3.10: Temporal analysis of different metrics for the refueling port in the *test_video_lab_platform_6* video. The metrics include (a) central position, (b) velocity, (c) acceleration, and (d) area, providing a comprehensive overview of the object's dynamics over time.

3.1.8 Framework Design Overview

The proposed framework (see figure 3.11) aims to predict the future bounding boxes of the refueling port over the next m frames of a video stream, using the observed bounding boxes from the current and previous frames. The observed bounding box at time step t is represented as $\mathbf{b}_t = [x_t, y_t, w_t, h_t]$, where (x_t, y_t) denotes the coordinates of the bounding box center, and w_t and h_t represent the width and height of the bounding box, respectively. The objective is to predict a sequence of future bounding boxes $\mathbf{S}_{t+1:t+m} = \{\mathbf{b}_{t+1}, \ldots, \mathbf{b}_{t+m}\}$ on the following m frames, based on the past n observed bounding boxes $\mathbf{S}_{t-1+n:t} = \{\mathbf{b}_{t-1+n}, \ldots, \mathbf{b}_t\}$.

The frame process begins with the input video sequence, which is analysed frame by frame. An object detection model identifies and locates the bounding boxes of the supply port. These bounding boxes are then used in subsequent stages of data processing.

Two key input vectors are extracted from the bounding box information: the *spatial dynamics vector* and the *dimensional attributes vector*. The spatial dynamics vector includes information about the position, velocity, and acceleration of the bounding box center, represented as $\mathbf{p}_t = [x_t, y_t, v_{x,t}, v_{y,t}, a_{x,t}, a_{y,t}]$. The dimensional attributes vector captures the size of the bounding box and the changes in its dimensions over time, expressed as $\mathbf{d}_t = [w_t, h_t, \Delta w_t, \Delta h_t]$.

These input vectors are fed into a sequence model, where they are processed through separate encoders to extract meaningful temporal features. The encoded features are then concatenated and passed to a decoder, which predicts the future bounding boxes for the refueling port across the next N frames. The resulting output, $\mathbf{B}_t = \{\mathbf{b}_{t+1}, \ldots, \mathbf{b}_{t+N}\}$, provides an estimate of the bounding box positions and sizes in the subsequent frames.

This method effectively leverages both the spatial and dimensional information of the bounding boxes to generate accurate predictions of future positions and sizes, which are essential for tasks such as object tracking and navigation.

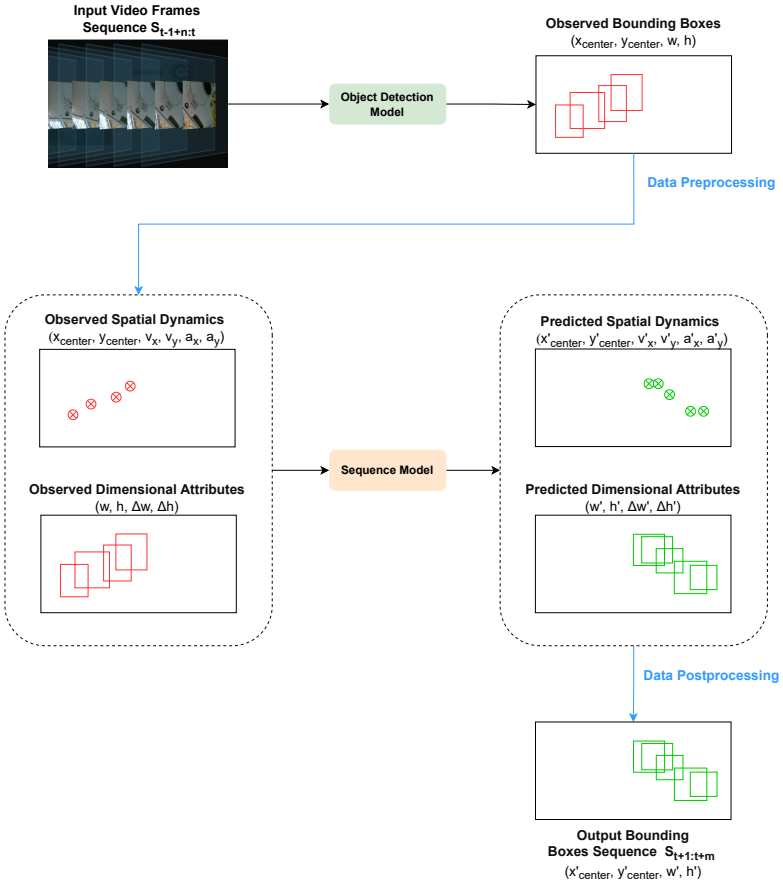


Figure 3.11: Framework Workflow

The proposed framework consists of two main components: an object detection model and a sequence model. The object detection model is responsible for detecting the refueling port in each frame of the video, while the sequence model predicts the future positions and sizes of the detected refueling port. The framework workflow is illustrated in Figure 3.11.

3.1.9 Object Detection Model

The object detection model provides a sequence of bounding boxes in each video frame, represented in the following format:

$$(x_{\text{center}}, y_{\text{center}}, w, h),$$

where $(x_{\text{center}}, y_{\text{center}})$ denotes the coordinates of the bounding box center, and (w, h) represent the width and height of the bounding box, respectively.

3.2 Sequence Model Design

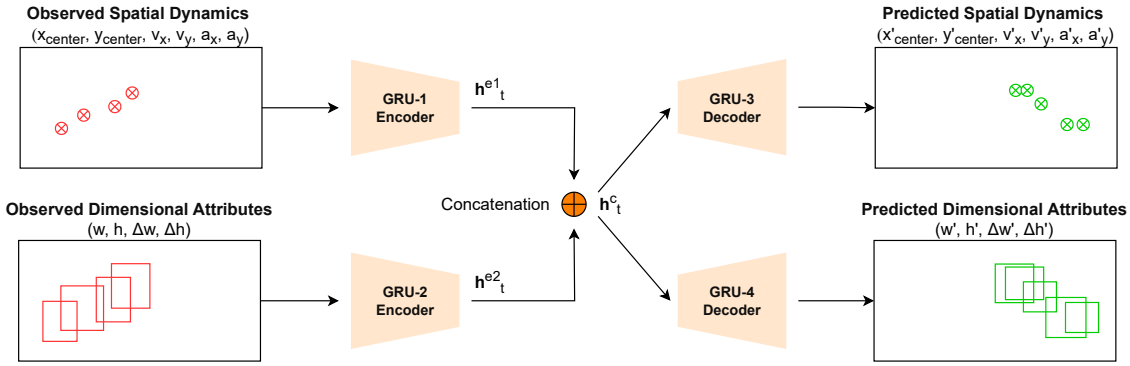


Figure 3.12: SizPos-GRU Model Architecture

This thesis introduces the SizPos-GRU model, a deep learning sequence model for predicting future positions and sizes of a unique object in a video stream. The model leverages an encoder-attention-decoder architecture to capture temporal dependencies and spatial relationships effectively. The framework consists of four main components: two encoders, an attention mechanism, and two decoders. Each component is designed to handle specific aspects of the sequence modeling task, from encoding input sequences to generating context-aware predictions.

3.2.1 Input Representation

The *SizPos-GRU* model expects two types of input sequences to predict the future positions and sizes of an object. The first input captures the spatial dynamics of the object, including its position, velocity and acceleration. The second input focuses on the object's dimensional attributes, such as its width and height, and changes in these dimensions over time.

3.2.1.1 Spatial Dynamics Vector

The spatial dynamics vector at time t, denoted as $\mathbf{p}_t$, is obtained by computing the position, velocity, and acceleration of the bounding box center:

$$\mathbf{p}_t = (x_t, y_t, v_{x,t}, v_{y,t}, a_{x,t}, a_{y,t}), \tag{3.1}$$

$$v_{x,t} = x_t - x_{t-1}, \quad v_{y,t} = y_t - y_{t-1}, \tag{3.2}$$

$$a_{x,t} = v_{x,t} - v_{x,t-1}, \quad a_{y,t} = v_{y,t} - v_{y,t-1}. \tag{3.3}$$

Here, (x_t, y_t) represents the center coordinates at time t, $(v_{x,t}, v_{y,t})$ are the velocities, and $(a_{x,t}, a_{y,t})$ are the accelerations along the x and y axes.

3.2.1.2 Dimensional Attributes Vector

The dimensional attributes vector at time t, denoted as $\mathbf{d}_t$, captures the size of the bounding box and the changes in its dimensions:

$$\mathbf{d}_t = (w_t, h_t, \Delta w_t, \Delta h_t), \tag{3.4}$$

$$\Delta w_t = w_t - w_{t-1}, \quad \Delta h_t = h_t - h_{t-1}. \tag{3.5}$$

Here, (w_t, h_t) represents the width and height at time t, and $(\Delta w_t, \Delta h_t)$ are the changes in these dimensions from the previous time step.

3.2.1.3 Input Sequences for Model

The sequences of these vectors over a time window T are fed into the model as:

$$\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_T\}, \quad \mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_T\},$$

where $\mathbf{P}$ is the sequence of spatial dynamics vectors and $\mathbf{D}$ is the sequence of dimensional attributes vectors. These sequences are then processed by the model's encoders to extract temporal features for predicting future bounding box positions and sizes.

These sequences are fed into the encoders to extract meaningful temporal features.

3.2.1.4 Encoders

The framework employs two separate LSTM encoders: one for processing the sequence of bounding boxes and another for processing the sequence of velocities. Each encoder is designed to capture the temporal dependencies within its respective input data.

The bounding box encoder is defined as:

$$H_p, (h_{p_n}, c_{p_n}) = \text{LSTM}_p(\mathbf{P}) \tag{3.6}$$

where $\mathbf{P} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_T)$ is the input sequence of bounding boxes, $H_p = (h_{p_1}, h_{p_2}, \ldots, h_{p_T})$ represents the sequence of hidden states, and (h_{p_n}, c_{p_n}) are the final hidden and cell states of the bounding box encoder.

The velocity encoder is defined as:

$$H_v, (h_{v_n}, c_{v_n}) = \text{LSTM}_v(\mathbf{V}) \tag{3.7}$$

where $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_T)$ is the input sequence of velocities, $H_v = (h_{v_1}, h_{v_2}, \ldots, h_{v_T})$ represents the sequence of hidden states, and (h_{v_n}, c_{v_n}) are the final hidden and cell states of the velocity encoder.

3.2.1.5 Attention Mechanism

The attention mechanism is crucial for allowing the model to focus on relevant parts of the input sequences when making predictions. It computes a set of attention weights that highlight important hidden states from both encoders. These weights are used to form context vectors, which provide the decoders with relevant information from the input sequences.

The attention weights α_t are calculated as follows:

$$\alpha_t = \text{softmax}(v^\top \tanh(W_{h_p} H_p + W_{h_v} H_v + W_s s_{t-1})) \tag{3.8}$$

where W_{h_p} and W_{h_v} are learned weight matrices for the bounding box and velocity hidden states, respectively, W_s is a learned weight matrix for the previous decoder state, and v is a learned vector. The context vectors c_{p_t} and c_{v_t} are then computed as:

$$c_{p_t} = \sum_{i=1}^{T} \alpha_{t,i} h_{p_i} \tag{3.9}$$

$$c_{v_t} = \sum_{i=1}^{T} \alpha_{t,i} h_{v_i} \tag{3.10}$$

These context vectors are passed to the decoders to generate the next time step predictions.

3.2.2 Decoders

The framework employs two separate LSTM decoders: one for generating predictions for future bounding boxes and another for generating predictions for future velocities. Each decoder takes as input the previous output and the context vectors from the attention mechanism.

The bounding box decoder's input at each time step t is the concatenated vector of the previous output and the context vector from the bounding box encoder:

$$\mathbf{d}_{p_t} = [\mathbf{p}_{t-1}, c_{p_{t-1}}] \tag{3.11}$$

The LSTM in the bounding box decoder processes this input to update its hidden and cell states:

$$s_{p_t}, (h_{p_t}, c_{p_t}) = \text{LSTM}_p(\mathbf{d}_{p_t}, (h_{p_{t-1}}, c_{p_{t-1}})) \tag{3.12}$$

The final output for the bounding box is obtained through a fully connected layer with a sigmoid activation function to ensure the output values are between 0 and 1:

$$\mathbf{p}_t = \text{Sigmoid}(\text{Linear}_p(s_{p_t})) \tag{3.13}$$

Similarly, the velocity decoder's input at each time step t is the concatenated vector of the previous output and the context vector from the velocity encoder:

$$\mathbf{d}_{v_t} = [\mathbf{v}_{t-1}, c_{v_{t-1}}] \tag{3.14}$$

The LSTM in the velocity decoder processes this input to update its hidden and cell states:

$$s_{v_t}, (h_{v_t}, c_{v_t}) = \text{LSTM}_v(\mathbf{d}_{v_t}, (h_{v_{t-1}}, c_{v_{t-1}})) \tag{3.15}$$

The final output for the velocity is obtained through a fully connected layer without any activation function. This design choice allows the output values to range freely, including negative values, which is necessary because velocities can be positive or negative:

$$\mathbf{v}_t = \text{Linear}_v(s_{v_t}) \tag{3.16}$$

The decoders continue this process for the entire prediction horizon, using their own previous outputs as inputs for future predictions.

3.2.3 Transformer-Based Architecture

The proposed model leverages a Transformer architecture to predict future positions and velocities of objects in a video stream. Transformers, introduced by Vaswani et al. (2017), have demonstrated superior performance in sequence-to-sequence tasks by effectively capturing long-range dependencies through self-attention mechanisms. Our architecture consists of three main components: an encoder, a decoder, and a multi-head self-attention mechanism. Each component is designed to handle specific aspects of the sequence modeling task, from encoding input sequences to generating context-aware predictions.

3.2.3.1 Input Representation

The model takes as input sequences of bounding boxes and their velocities extracted from video frames. Each bounding box is represented by its center coordinates $(x_{\text{center}}, y_{\text{center}})$, width (w), and height (h). Corresponding velocities are represented by changes in these values: $(\Delta x, \Delta y, \Delta w, \Delta h)$. Thus, each input vector $\mathbf{p}_t$ and $\mathbf{v}_t$ at time t can be represented as:

$$\mathbf{p}_t = (x_{\text{center},t}, y_{\text{center},t}, w_t, h_t) \tag{3.17}$$

$$\mathbf{v}_t = (\Delta x_t, \Delta y_t, \Delta w_t, \Delta h_t) \tag{3.18}$$

These sequences are fed into the encoder to extract meaningful temporal features.

3.2.3.2 Encoder

The encoder consists of multiple layers, each containing two main components: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. The input to the encoder is a sequence of bounding box and velocity vectors. The encoder processes these sequences independently using the following steps:

- **Positional Encoding**: Since the Transformer model does not have a built-in notion of order, positional encodings are added to the input embeddings to incorporate sequence order information.
- **Self-Attention**: The multi-head self-attention mechanism allows the model to focus on different parts of the input sequence simultaneously, capturing dependencies regardless of their distance in the sequence.
- **Feed-Forward Network**: A position-wise fully connected feed-forward network is applied to the output of the self-attention mechanism to introduce non-linearity and further process the information.
- **Residual Connection and Layer Normalization**: Residual connections around each sub-layer followed by layer normalization help in stabilizing the training and improving gradient flow.

3.2.3.3 Decoder

The decoder is similarly composed of multiple layers, with each layer containing three main components: a multi-head self-attention mechanism, a multi-head encoder-decoder attention mechanism, and a position-wise fully connected feed-forward network. The decoder generates predictions for future bounding boxes and velocities using the following steps:

- **Masked Self-Attention**: The decoder's self-attention mechanism is masked to prevent positions from attending to subsequent positions, ensuring that the prediction for a position depends only on known outputs.
- **Encoder-Decoder Attention**: This mechanism allows the decoder to focus on relevant parts of the encoder's output, providing context from the input sequence.
- **Feed-Forward Network**: Similar to the encoder, a fully connected feed-forward network is applied to the output of the attention mechanisms.

- **Residual Connection and Layer Normalization**: As in the encoder, residual connections and layer normalization are applied to stabilize training and improve gradient flow.

3.2.3.4 Multi-Head Self-Attention

The multi-head self-attention mechanism is a key component of the Transformer architecture, enabling the model to jointly attend to information from different representation subspaces. It works by projecting the input sequence into multiple sets of queries, keys, and values, performing scaled dot-product attention for each set, and concatenating the results. This process allows the model to capture various aspects of the input sequence, enhancing its ability to learn complex patterns.

3.2.3.5 Output Generation

The output of the decoder is passed through a linear layer to generate predictions for the bounding box coordinates and velocities. Separate linear layers are used for bounding box predictions (with a sigmoid activation to ensure values are between 0 and 1) and velocity predictions (without activation to allow for positive and negative values).

3.2.3.6 Loss Function and Training

The model is trained to minimize the mean squared error between the predicted and actual bounding boxes and velocities. The loss function is defined as:

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[(x_{i,t}-\hat{x}_{i,t})^2+(y_{i,t}-\hat{y}_{i,t})^2+(w_{i,t}-\hat{w}_{i,t})^2+(h_{i,t}-\hat{h}_{i,t})^2+(\Delta x_{i,t}-\hat{\Delta x}_{i,t})^2+(\Delta y_{i,t}-\hat{\Delta y}_{i,t})^2+(\right. \tag{3.19}$$

The Adam optimizer is used to update the model parameters, and early stopping based on validation loss is employed to prevent overfitting and ensure generalization.

3.3 Algorithm Design

3.4 Data Augmentation Strategy

The data augmentation strategy is designed to enhance the robustness and generalization capability of the model. This strategy simulates various camera movements and imperfections that might occur in real-world scenarios. The augmentation approach consists of the following key components:

3.4.1 Sequence Reversal

For the training stage, the size of the dataset is doubled by including reversed sequences:

- For each original sequence $(S_1, S_2, ..., S_n)$, a reversed sequence $(S_n, S_{n-1}, ..., S_1)$ is added.
- This augmentation helps the model learn to predict both forward and backward movements of the camera relative to the aircraft refueling port.

3.4.2 Camera Movement Simulation

Subtle camera movements are simulated to make the model more robust to varying camera positions:

- **Panning**: A smooth, cumulative random offset is applied to the x and y coordinates of the bounding boxes. This simulates the effect of the camera panning horizontally or vertically.
- The panning effect is modeled as:

$$\text{pan}_t = \sum_{i=1}^{t} \frac{N(0, \sigma^2)}{5} \tag{3.20}$$

  where $N(0, \sigma^2)$ is a normal distribution with mean 0 and variance σ^2.

3.4.3 Zoom Simulation

To account for changes in the apparent size of the refueling port due to camera zoom or aircraft movement:

- A smooth, cumulative scaling factor is applied to the width and height of the bounding boxes.
- The zoom effect is modeled as:

$$\text{zoom}_t = \prod_{i=1}^{t} U(0.98, 1.02) \tag{3.21}$$

  where $U(0.98, 1.02)$ is a uniform distribution between 0.98 and 1.02.

3.4.4 Detection Inaccuracy Simulation

To make the model more robust to slight inaccuracies in object detection:

- Small Gaussian noise is added to all bounding box coordinates.
- The noise is sampled from $N(0, 0.002^2)$ for each coordinate.

3.4.5 Implementation Details

The augmentation strategy is applied with the following considerations:

- Augmentations are only applied during the training stage.
- Each sequence has a 50% chance of being augmented.
- When applied, augmentations affect both input and output sequences consistently to maintain temporal coherence.
- All augmented values are clipped to the range [0, 1] to ensure they remain valid normalized coordinates in the YOLO format.

This comprehensive augmentation strategy enhances the diversity of the training data, helping the model generalize better to various real-world scenarios involving camera movements and detection variations.

Chapter 4

Experiment Design

4.1 Experiment Environment

4.2 Comparison Experiments

4.3 Evaluation Metrics

To evaluate the performance of the proposed framework, four key metrics are used: Average Displacement Error (ADE), Final Displacement Error (FDE), Average Intersection over Union (AIOU), and Final Intersection over Union (FIOU). These metrics provide a comprehensive assessment of the accuracy and robustness of the predicted bounding boxes over time.

Average Displacement Error (ADE)

The Average Displacement Error (ADE) measures the average Euclidean distance between the predicted bounding box centers and the ground truth bounding box centers over all predicted frames. It is defined as:

$$\text{ADE} = \frac{1}{m}\sum_{t=1}^{m}\sqrt{(x_t^{\text{pred}} - x_t^{\text{gt}})^2 + (y_t^{\text{pred}} - y_t^{\text{gt}})^2}, \tag{4.1}$$

where m is the number of predicted frames, $(x_t^{\text{pred}}, y_t^{\text{pred}})$ represents the center of the predicted bounding box at the t-th frame, and $(x_t^{\text{gt}}, y_t^{\text{gt}})$ represents the center of the ground truth bounding box at the same frame.

Final Displacement Error (FDE)

The Final Displacement Error (FDE) focuses on the accuracy of the predicted bounding box center in the final frame of the prediction horizon. It is defined as the Euclidean distance between the predicted and ground truth bounding box centers at the final frame m:

$$\text{FDE} = \sqrt{(x_T^{\text{pred}} - x_T^{\text{gt}})^2 + (y_T^{\text{pred}} - y_T^{\text{gt}})^2}. \tag{4.2}$$

Average Intersection over Union (AIOU)

The Average Intersection over Union (AIOU) evaluates the overlap between the predicted bounding boxes and the ground truth bounding boxes, averaged over all predicted frames. It is computed as:

$$\text{AIOU} = \frac{1}{m} \sum_{t=1}^{m} \frac{A(\mathbf{b}_t^{\text{pred}} \cap \mathbf{b}_t^{\text{gt}})}{A(\mathbf{b}_t^{\text{pred}} \cup \mathbf{b}_t^{\text{gt}})}, \tag{4.3}$$

where $A(\mathbf{b}_t^{\text{pred}} \cap \mathbf{b}_t^{\text{gt}})$ denotes the area of the intersection between the predicted bounding box $\mathbf{b}_t^{\text{pred}}$ and the ground truth bounding box $\mathbf{b}_t^{\text{gt}}$ at the t-th frame, and $A(\mathbf{b}_t^{\text{pred}} \cup \mathbf{b}_t^{\text{gt}})$ denotes the area of their union.

Final Intersection over Union (FIOU)

The Final Intersection over Union (FIOU) assesses the overlap between the predicted bounding box and the ground truth bounding box at the final frame of the prediction horizon. It is defined as:

$$\text{FIOU} = \frac{A(\mathbf{b}_T^{\text{pred}} \cap \mathbf{b}_T^{\text{gt}})}{A(\mathbf{b}_T^{\text{pred}} \cup \mathbf{b}_T^{\text{gt}})}, \tag{4.4}$$

where $A(\mathbf{b}_T^{\text{pred}} \cap \mathbf{b}_T^{\text{gt}})$ and $A(\mathbf{b}_T^{\text{pred}} \cup \mathbf{b}_T^{\text{gt}})$ denote the intersection and union areas at the final frame m, respectively.

These metrics provide a balanced evaluation of the prediction performance, capturing both the spatial accuracy and the temporal consistency of the predicted bounding boxes.

Chapter 5

Results and Discussion

5.1 Object Detection Training Results

This section presents the final testing results of three different YOLO models (yolov10n.pt, yolov10s.pt, yolov10m.pt) on a dataset involving the classification of fuel port states (CLOSED, SEMI-OPEN, OPEN).

Table 5.1: Comparison of YOLO Models

Model	Metric	All	Fuel Port [CLOSED]	Fuel Port [SEMI-OPEN]	Fuel Port [OPEN]
YOLOv10n.pt	Precision (P)	0.989	0.994	0.982	0.992
	Recall (R)	0.876	0.657	0.996	0.974
	mAP50	0.893	0.696	0.995	0.987
	mAP50-95	0.852	0.684	0.962	0.909
YOLOv10s.pt	Precision (P)	0.997	0.998	1.000	0.992
	Recall (R)	0.884	0.682	0.997	0.974
	mAP50	0.893	0.694	0.995	0.989
	mAP50-95	0.848	0.678	0.961	0.905
YOLOv10m.pt	Precision (P)	0.994	0.996	0.999	0.988
	Recall (R)	0.888	0.682	1.000	0.983
	mAP50	0.892	0.701	0.995	0.981
	mAP50-95	0.852	0.692	0.968	0.897

5.1.1 Summary

Each model demonstrates strengths in different aspects of performance metrics. The YOLOv10s.pt model exhibits the highest precision at 0.997, while the YOLOv10m.pt model shows the highest recall at 0.888. The YOLOv10n.pt and YOLOv10m.pt models have the highest mAP50-95 at 0.852. The selection of an optimal model should consider the specific requirements of precision, recall, and mAP for the intended application.

5.2 Data Description

5.3 Experiment Results

5.4 Testing Visualisation

Chapter 6

Conclusion and Future Work

References

1. A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. volume 2016-December, 2016. doi: 10.1109/CVPR.2016.110.

2. A. F. C. A. C. A. . M. C. (AvMC). Ar3p program background (2017-2019) and robotic hot refueling demonstrations (sep-oct 2020), 2020. URL hhttps://www.denix.osd.mil/ndcee/denix-files/sites/44/2023/09/EXSUM-AR3P-Robotic-Refueling-K-MAX-and-S-70-Sep_Oct-2020-v9.pdf. Accessed: 2024-07-24.

3. R. A. Bennett, Y. C. Shiu, and M. B. Leahy. A robust light invariant vision system for aircraft refueling. volume 1, 1991. doi: 10.1109/robot.1991.131568.

4. S. Blakey, L. Rye, and C. W. Wilson. Aviation gas turbine alternative fuels: a review. *Proceedings of the Combustion Institute*, 33(2):2863–2885, 2011. doi: 10.1016/j.proci.2010.09.011.

5. A. Bochkovskiy, C. Wang, and H. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020. URL https://arxiv.org/abs/2004.10934.

6. H. Burnette. Lab demonstrates robotic ground refueling of aircraft, Oct. 2010. URL https://www.wpafb.af.mil/News/Article-Display/Article/400022/lab-demonstrates-robotic-ground-refueling-of-aircraft/. Accessed: 2024-07-24.

7. F. Chen, X. Wang, Y. Zhao, S. Lv, and X. Niu. Visual object tracking: A survey. *Computer Vision and Image Understanding*, 222, 9 2022. ISSN 1090235X. doi: 10.1016/j.cviu.2022.103508.

8. Y. Chen, X. Yuan, R. Wu, J. Wang, Q. Hou, and M.-M. Cheng. Yolo-ms: Rethinking multi-scale representation learning for real-time object detection, 2023.

9. G. Cheng, X. Yuan, X. Yao, K. Yan, Q. Zeng, X. Xie, and J. Han. Towards large-scale small object detection: Survey and benchmarks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45, 2023. ISSN 19393539. doi: 10.1109/TPAMI.2023.3290594.

10. O. Cokorilo, S. Gvozdenovic, L. Vasov, and P. Mirosavljevic. Costs of unsafety in aviation. *Technological and Economic Development of Economy - TECHNOL ECON DEV ECON*, 16:188–201, 06 2010. doi: 10.3846/tede.2010.12.

11. J. Dai, Y. Li, K. He, and J. Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016. URL http://arxiv.org/abs/1605.06409.

12. A. Eitel, N. Hauff, and W. Burgard. Learning to singulate objects using a push proposal network. *CoRR*, abs/1707.08101, 2017. URL http://arxiv.org/abs/1707.08101.

13. H. Face. Object detection leaderboard, 2023. URL https://huggingface.co/blog/object-detection-leaderboard. Accessed: 2024-06-17.

14. H. Fan, L. Zhu, and Y. Yang. Cubic lstms for video prediction. page 8263 – 8270, 2019. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85085021761&partnerID=40&md5=5ef1e8834ec46834bbb53bbcf11720c9. Cited by: 41.

15. N. Ficken. Concept demonstration explores robotic aviation refueling system, July 18 2017. URL https://www.army.mil/article/190980/concept_demonstration_explores_robotic_aviation_refueling_system. Accessed: 2024-07-24.

16. C. Finn, I. J. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. *CoRR*, abs/1605.07157, 2016. URL http://arxiv.org/abs/1605.07157.

17. Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun. YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430, 2021. URL https://arxiv.org/abs/2107.08430.

18. R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL http://arxiv.org/abs/1504.08083.

19. R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL http://arxiv.org/abs/1311.2524.

20. K. Gong, B. Liu, X. Xu, Y. Xu, Y. He, Z. Zhang, and J. Rasol. Research of an unmanned aerial vehicle autonomous aerial refueling docking method based on binocular vision. *Drones*, 7, 2023. ISSN 2504446X. doi: 10.3390/drones7070433.

21. Y. Hong, H. Martin, and M. Raubal. How do you go where?: improving next location prediction by learning travel mode information using transformers. 2022. doi: 10.1145/3557915.3560996.

22. H. Huang and Y. Guan. A dual-branch spatial-temporal learning network for video prediction. *IEEE Access*, 12:73258 – 73267, 2024. doi: 10.1109/ACCESS.2024.3394209. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85191878045&doi=10.1109%2fACCESS.2024.3394209&partnerID=40&md5=a2491ede039a7189014acb3b7eb0256d. Cited by: 0; All Open Access, Gold Open Access.

23. Intel. Intel realsense depth camera d435, 2024. URL https://www.intelrealsense.com/depth-camera-d435/. Accessed: 2024-06-24.

24. B. Jin, X. Song, J. Li, and P. Zhang. Unsupervised video forecasting with flow parsing mechanism of human visual system. *Engineering Applications of Artificial Intelligence*, 134:108652, 2024. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2024.108652. URL https://www.sciencedirect.com/science/article/pii/S0952197624008108.

25. G. Jocher. Yolov5 release v7.0, 2022. URL https://github.com/ultralytics/yolov5/tree/v7.0.

26. G. Jocher. Yolov8, 2023. URL https://github.com/ultralytics/ultralytics/tree/main.

27. J. Kang, S. Tariq, H. Oh, and S. Woo. A survey of deep learning-based object detection methods and datasets for overhead imagery. *IEEE Access*, 10:1–1, 01 2022. doi: 10.1109/ACCESS.2022.3149052.

28. M. M. Karim, R. Qin, and Y. Wang. Fusion-gru: A deep learning model for future bounding box prediction of traffic agents in risky driving videos. *Transportation Research Record*, 2024. ISSN 21694052. doi: 10.1177/03611981241230540.

29. B. Kuang, S. Barnes, G. Tang, and K. Jenkins. A dataset for autonomous aircraft refueling on the ground (agr). 2023. doi: 10.1109/ICAC57885.2023.10275212.

30. J. Kugarajeevan, T. Kokul, A. Ramanan, and S. Fernando. Transformers in single object tracking: An experimental survey. *IEEE Access*, 11, 2023. ISSN 21693536. doi: 10.1109/ACCESS.2023.3298440.

31. W. C. Lee, Y. B. Jeon, S. S. Han, and C. S. Jeong. Position prediction in space system for vehicles using artificial intelligence. *Symmetry*, 14, 2022. ISSN 20738994. doi: 10.3390/sym14061151.

32. C. Li, L. Li, Y. Geng, H. Jiang, M. Cheng, B. Zhang, Z. Ke, X. Xu, and X. Chu. Yolov6 v3.0: A full-scale reloading, 2023.

33. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, 2018.

34. K. Liu, L. Chen, and X. Liu. Research on application of frontier technologies at smart airport. In J. Zeng, P. Qin, W. Jing, X. Song, and Z. Lu, editors, *Data Science*, pages 319–330, Singapore, 2021. Springer Singapore. ISBN 978-981-16-5943-0.

35. S. Liu, D. Liu, G. Srivastava, D. Połap, and M. Woźniak. Overview and methods of correlation filter algorithms in object tracking. *Complex and Intelligent Systems*, 7, 2021. ISSN 21986053. doi: 10.1007/s40747-020-00161-4.

36. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. URL http://arxiv.org/abs/1512.02325.

37. M. D. L. Olja Čokorilo and G. Dell'Acqua. Aircraft safety analysis using clustering algorithms. *Journal of Risk Research*, 17(10):1325–1340, 2014. doi: 10.1080/13669877.2013.879493. URL https://doi.org/10.1080/13669877.2013.879493.

38. E. Plaza and M. Santos. Knowledge based approach to ground refuelling optimization of commercial airplanes. *Expert Systems*, 38, 2021. ISSN 14680394. doi: 10.1111/exsy.12631.

39. N. Ranjan, S. Bhandari, Y.-C. Kim, and H. Kim. Video frame prediction by joint optimization of direct frame synthesis and optical-flow estimation. *Computers, Materials and Continua*, 75(2):2615 – 2639, 2023. doi: 10.32604/cmc.2023.026086. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85152478811&doi=10.32604%2fcmc.2023.026086&partnerID=40&md5=3c3a7145b11a082102840f176198a13d. Cited by: 0; All Open Access, Gold Open Access.

40. J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016. URL http://arxiv.org/abs/1612.08242.

41. J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL http://arxiv.org/abs/1506.02640.

42. S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2577031.

43. M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. doi: 10.1109/CVPR.2008.4587727.

44. R. Sati, S. Singh, and R. Yadav. Aircraft fuel system: design, components, and safety. *International Journal of Aerospace Engineering*, 2019:1–12, 2019. doi: 10.1155/2019/6943787.

45. C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36 Vol.3, 2004. doi: 10.1109/ICPR.2004.1334462.

46. E. R. Schultz. Robotic systems for aircraft servicing/maintenance. *IEEE Aerospace and Electronic Systems Magazine*, 1, 1986. ISSN 08858985. doi: 10.1109/MAES.1986.5005018.

47. X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, abs/1506.04214, 2015. URL http://arxiv.org/abs/1506.04214.

48. N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, abs/1502.04681, 2015. URL http://arxiv.org/abs/1502.04681.

49. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin. Attention is all you need. volume 2017-December, 2017.

50. A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding. Yolov10: Real-time end-to-end object detection, 2024.

51. C. Wang, W. He, Y. Nie, J. Guo, C. Liu, K. Han, and Y. Wang. Gold-yolo: Efficient object detector via gather-and-distribute mechanism, 2023.

52. C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao. Yolov9: Learning what you want to learn using programmable gradient information, 2024.

53. X. Wang, X. Dong, X. Kong, J. Li, and B. Zhang. Drogue detection for autonomous aerial refueling based on convolutional neural networks. *Chinese Journal of Aeronautics*, 30, 2017. ISSN 10009361. doi: 10.1016/j.cja.2016.12.022.

54. J. Wu and S. Xu. From point to region: Accurate and efficient hierarchical small object detection in low-resolution remote sensing images. *Remote Sensing*, 13, 2021. ISSN 20724292. doi: 10.3390/rs13132620.

55. S. Xu, X. Wang, W. Lv, Q. Chang, C. Cui, K. Deng, G. Wang, Q. Dang, S. Wei, Y. Du, and B. Lai. Pp-yoloe: An evolved version of yolo, 2022.

56. X. Xu, Y. Jiang, W. Chen, Y. Huang, Y. Zhang, and X. Sun. Damo-yolo : A report on real-time object detection design, 2023.

57. T. Ye, W. Qin, Z. Zhao, X. Gao, X. Deng, and Y. Ouyang. Real-time object detection network in uav-vision based on cnn and transformer. *IEEE Transactions on Instrumentation and Measurement*, 72, 2023. ISSN 15579662. doi: 10.1109/TIM.2023.3241825.

58. S. Yildirim, Z. Rana, and G. Tang. Autonomous ground refuelling approach for civil aircrafts using computer vision and robotics. volume 2021-October, 2021. doi: 10.1109/DASC52595.2021.9594312.

59. S. Yildirim, Z. A. Rana, and G. Tang. Development of vision guided real-time trajectory planning system for autonomous ground refuelling operations using hybrid dataset. 2023. doi: 10.2514/6.2023-1148.

60. S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee. A survey of modern deep learning based object detection models, 2022. ISSN 10512004.

61. Y. Zhang, T. Wang, K. Liu, B. Zhang, and L. Chen. Recent advances of single-object tracking methods: A brief survey. *Neurocomputing*, 455, 2021. ISSN 18728286. doi: 10.1016/j.neucom.2021.05.011.

62. Z. Zhong, D. Li, H. Wang, and Z. Su. Drogue position and tracking with machine vision for autonomous air refueling based on ekf. volume 2, 2017. doi: 10.1109/IHMSC.2017.151.